Content Protection for Optical Media

A Comparison of Self-Protecting Digital Content and AACS

Independent Security Evaluators
www.securityevaluators.com

May 3, 2005

Executive Summary

This report details the findings of Independent Security Evaluators in evaluating the Cryptography Research Inc. **Self-Protecting Digital Content** (SPDC) specification. Our evaluation considered VM specification version 1.0b and Interface Specification 0.95a. We also examined the integration of SPDC with the Advanced Access Content System (AACS) as defined by the AACS public specification revision 0.90 (dated April 14, 2005), along with supplemental slides and materials. We first note several flaws in the current version of the AACS specification, and further examine the implications of device compromise on the security offered by AACS. We conclude that in many instances the SPDC framework offers significant advantages in content protection over AACS alone.

Self-Protecting Digital Content. The focus of this report is on the use of SPDC as a core component of a secure digital content distribution system. SPDC provides content creators with a means to author title-specific "content code" for execution at the device. This content code allows for the software re-implementation of player processing logic, in the event that class of devices is compromised or is found to contain exploitable faults. This provides a means to enable renewability even where device hardware cannot be fully trusted.

We compare the security offered by SPDC/AACS to the security provided by AACS alone, in scenarios involving device and device-class compromise (e.g., hardware compromise of a single player, or compromise of all players of a given model.) We note that SPDC provides immediate content protection in a variety of compromise scenarios, whereas AACS achieves protection only for future titles (via device revocation). Furthermore, even when considering protection for future titles, device revocation schemes such as AACS do not adequately address the case of device-class compromise.

No device-side content protection system can withstand all attacks. We note explicitly that the protection offered by SPDC content code should be considered time-limited. The principal advantage of SPDC is the ability to recover system security, potentially with each new title released. SPDC also provides mechanisms by which content creators can overcome or repair device implementation flaws, and provides a generally more resilient level of system security than can be provided by key management and device-revocation systems alone.

Contents

Introduction

This report details the findings of ISE in evaluating the Cryptography Research Inc. **Self-Protecting Digital Content** (SPDC) architecture. Our evaluation considered the VM Specification version 1.0b and Interface Specification 0.95a. We also examined the integration of SPDC with the Advanced Access Content System (AACS) as defined by the AACS public specification revision 0.90 (dated April 14, 2005), along with supplemental slides and materials.

The focus of this report is the use of SPDC as a core component of a secure digital content distribution system. SPDC provides content creators with a means to author title-specific content code for execution on a standardized Virtual Machine (VM) with a sufficient instruction set and a device API (interface). These allow for the software re-implementation of player processing logic, in the event that class of devices is compromised or is found to contain exploitable faults. This provides for renewability even in the case that device hardware cannot be fully trusted.

Report Organization. This report is organized as follows. Below, we provide a high-level overview of content protection techniques and SPDC. The next section consists of a detailed security analysis of AACS. Finally, we propose a full attack model for digital content protection systems, and compare the security properties of AACS against a combined system including both AACS and SPDC components.

Overview: Protecting Digital Content

Device Revocation Schemes. Revocation provides a means to achieve continued content protection in the face of device compromise or unauthorized content distribution. Unlike the Content Scrambling System (CSS), which provides the same decryption keys to every player of a given model, AACS provisions each individual player with a unique set of decryption keys. Under normal circumstances, any set of keys can be used to correctly decrypt published content. However, in the event that content producers detect an unauthorized use of device keys, AACS provides the ability to revoke any set of keys, by ensuring that they cannot be used to decrypt future titles. This provides an effective means to permanently disable any device keys that are used in unauthorized manner, e.g., published as part of an illegal software player, or used to perform unauthorized content decryption.

In order for device-by-device revocation to be practical, two conditions must be met:

1. **Individual device compromise must be rare and/or costly to the adversary.** In practice, this is difficult to guarantee. Once an adversary has devel-

oped a successful attack technique on a specific player model, the cost of repeating the compromise may drop significantly. If a resourceful, financially motivated adversary can gain an advantage against the protection system by compromising multiple devices, it is logical to assume that he will seek to do so. Alternatively, an attacker who is not motivated by profit is likely to disseminate the details of a successful attack to the general public, resulting in a high likelihood that the attack will be replicated.

2. **Content creators must know which devices to revoke.** In order to revoke a player used to decode pirated content or recover keys, content revocation schemes must incorporate a means by which content producers may *trace* a player based on its output. If an attacker can prevent content creators from successfully tracing a compromised player, then the revocation scheme is effectively neutralized. Therefore it is critical that any device revocation scheme incorporate an effective tracing mechanism that can survive device compromise. The tracing scheme defined by AACS does not meet these requirements, as discussed in the AACS section of this report.

Shortcomings of Device Revocation Schemes. In some cases, widely disseminated and repeatable attacks may leave an entire class of devices (e.g., all players of a specific make and model) vulnerable to compromise. As discussed above, a widespread and repeatable compromise against a device class limits the effectiveness of device-by-device revocation, as an adversary can easily replace any specific device once it is revoked. Such a compromise may also reduce the effectiveness of other components of player logic, including tracing subsystems tasked with embedding forensic marks into decoded content (forensic marks are device-identifying marks which facilitate tracing of pirated content back to the device which produced it). This can further limit content producers' ability to trace and revoke compromised players.

Unfortunately, device revocation schemes offer a poor set of options in the event of an easily replicable compromise. Clearly, device-by-device revocation offers limited protection (and in the case of AACS, comes at some storage cost). To fully defend against this attack, content creators may be forced to revoke *all* devices from the given class, even those belonging to legitimate users. Such a drastic solution comes at a high cost, and may be financially infeasible for content producers and manufacturers to implement.

Title Revocation. Content protection schemes must also consider an adversary who duplicates encrypted media as is, rather than attacking the encryption or player tech-

nology. Protection systems can defend against unsophisticated duplication by mandating read-only elements on physical media, e.g., a disc serial number that cannot be altered by consumer devices. Unfortunately, such protection serves only to prevent duplication via consumer technology. An adversary who possesses sophisticated duplication technology might succeed in producing copies that devices cannot distinguish from an original title.

In the event that a title suffers widespread duplication, content producers may choose to *revoke* the title by issuing a Title Revocation message. Title Revocations must be transmitted to players via an alternate channel (e.g., by distribution on other non-revoked titles, or via Internet connectivity). Once received by a player, Revocations are typically stored in local non-volatile RAM. Prior to playback of any title, the device first checks for the presence of a Revocation in its storage. Behavior in the presence of a revoked title may vary: systems may simply terminate processing, or they may attempt to validate the title through some external mechanism (e.g., by connecting to the Internet, or by requesting entry of a code from the title's packaging). It is important to note that title revocation will disable playback of both legitimate and illegitimate copies of a title, and should therefore used only in extreme circumstances.

Self-Protecting Digital Content

SPDC is designed to provide an additional layer of security for a content protection system, complementing key management systems such as AACS. A primary goal of the SPDC framework is to provide renewability in the event that an entire class of devices is vulnerable to compromise.

Logic Renewal. SPDC provides a means for content creators to bypass compromised playback logic, by re-implementing the necessary functionality in software. This obfuscated software runs at the device, and can perform a variety of operations including player validation, state-keeping, forensic marking, and decryption. Furthermore, this software ("content code") can be unique for each title, which limits an adversary's ability to re-use a successful attack on one title in order to compromise future titles. This prevents single-title, or single-device attacks from escalating into a full-system compromise.

Targeted Renewal. On detection of a widespread device-class compromise, content code can be authored to perform actions when running on a device from the compromised class. Content creators can author new code specific to the compromised device-class, ceasing its ability to recover any digital content whatsoever from future title releases. Content code may also repair compromised playback logic by applying firmware patches, or it may limit attacks by requiring additional verification (e.g.,

authorization codes) before allowing playback. In this manner, content code may be used to recover from a device-class compromise without requiring mass revocation.

An adversary who possesses a fully compromised device might attempt to misrepresent the device identity to content code. In this case, content code should still have the means to detect, or "fingerprint" the underlying device through the execution of native code or by performing other measurements. Fingerprinting based on (but not limited to) encryption circuitry, memory access, or disc seek timings may also be used to differentiate between valid players and those that are malicious or tampered. Information derived from fingerprinting can be incorporated into content output through the use of a forensic marking scheme, giving content creators critical information about the nature of a device compromise.

Forensic Marking. Content protection schemes may use device-identifying forensic marks, which are embedded into player output, as a means to trace and revoke devices used to pirate content. Because these marks identify a given device, they must be generated within the device itself, and should survive attempts by an adversary to "scrub" marks from content, or bypass the marking system altogether. Unfortunately, marking schemes may fail against a motivated attacker who observes the output of many devices, or gains complete control over a device. In the event that a given marking scheme becomes ineffective, updated marking schemes can be implemented in content code and shipped with newer title releases. This forces adversaries to repeatedly invest effort to compromise marking, and prevents rapid mark removal on newer titles.

Device Storage. SPDC incorporates a mechanism for maintaining state on a local device. State may be used for title-specific purposes (e.g., recording user playback preferences), or as a means to implement additional protection mechanisms such as Title Revocation. SPDC defines a security mechanism, which effectively prevents unauthorized content code from accessing data stored in slots. These mechanisms are described later in the report.

Recognized Limitations of SPDC

The SPDC framework is intended to delay and frustrate an adversary's efforts to compromise digital content, but not necessarily to prevent such an effort completely. Most importantly, SPDC can be used to ensure that the effort spent in acquiring one title is not transferable to any future titles; with each attack the adversary must start anew. The motivation for this design is that the most significant financial damage suffered by the widespread piracy of digital content is inflicted within the first few weeks following a title's release. By delaying an adversary in compromising a given

title, the financial harm is minimized. While this represents a tradeoff, it offers benefits in situations such as the case of device-class compromise. In such cases, we believe that the protection provided by SPDC is substantial when considered against the cost (financial and otherwise) of revoking an entire class of devices.

Cost Considerations

A significant component in the decision to use SPDC—or any other content protection system—is the cost and complexity of maintaining the system. Because SPDC provides a significantly more flexible level of protection than systems such as AACS, development costs cannot be compared directly. For the purposes of this evaluation, SPDC is considered as an additional layer on top of other protection systems such as AACS. In the ideal case considered by the AACS standard (in which no widespread device compromise exists), SPDC content code development requires a very limited investment. In this case, it may be sufficient to deploy a single piece of content code across multiple titles, without the use of any device-specific functionality. In the absence of device-specific functionality, testing content code is necessary only to ensure that no device implementation suffers from significant implementation errors that would prevent it from executing standard code. The cost of this development and testing is minimal, as it can be amortized across the number of titles that deploy the content code.

In cases where it is desirable to deploy title-specific obfuscated content code, attempts to recover system security via AACS device revocation may have already failed. Consequently, the cost of deploying new content code must be weighed against the high costs of taking no action, or of revoking an entire device class.

AACS

This section presents the results of our analysis of the Advanced Access Content System (revision 0.90) based on the most recent specification documents, along with additional slides and materials. Our analysis considered the security of AACS as the sole component of a content protection system, particularly in the face of device and device-class compromise. In the course of our analysis, we discovered several potential security weaknesses in the AACS specification.

Overview of AACS. The Advanced Access Content System allows content producers to distribute content such that only a chosen subset of players may decrypt a given title. In practice, this approach can be used to permanently *revoke* specific devices (e.g., a compromised player)—effectively barring these devices from decrypting titles, and rendering them useful only for playback of content published prior to revocation. AACS also provides cryptographic mechanisms to detect content tampering; to revoke titles; and to trace pirated content and keys back to a compromised player. AACS also provides a number of enhanced features, including protocols for network communication.

Content in an AACS-enabled system is encrypted using a standard encryption scheme under one or more *Title Keys*. Title keys are themselves derived by combining volume and policy information with a corresponding set of *Media Keys*. To enable content decryption, AACS uses a broadcast encryption scheme to encipher Media Keys for distribution such that only non-revoked devices can successfully decrypt them. This effectively prevents revoked devices from accessing the encrypted content.

The form of broadcast encryption used by AACS is based on the stateless "Subset Difference" approach of Naor, Naor and Lotspiech. AACS broadcast encryption produces a *Media Key Block* (MKB) which is distributed along with the encrypted content. Every individual device in the system is pre-provisioned with a unique set of secret *Device Keys* which it may use to recover the Media Key from the MKB, provided that it has not been revoked. The broadcast encryption implemented by AACS does not require devices to keep state or participate honestly in the protocol.

General Concerns

The primary mechanism by which AACS renews system security is device revocation, which disables players in the event of compromise or misuse. This feature allows content producers to revoke any player that is used in an unauthorized manner, e.g. to decode content for illegal publication, or to extract decryption keys for use in a software player. We stipulate that while device revocation can successfully prevent a

device from decrypting titles, this protection applies only to future titles. In the next sections, we the protection provided by AACS in various circumstances of device and device-class compromise.

Insufficient Renewability and Robustness. While AACS offers strong cryptographic protection for content, it does not provide sufficient security as the sole component of a content protection scheme. This is particularly true when the goal is to prevent a spectrum of attacks aimed at compromising both players and protected content. For example, AACS is not designed to survive device-class attack scenarios, in which a replicable attack exists against a given player model.

Detection and Marking. Device revocation is a powerful tool for disabling devices that are known to be compromised. Unfortunately, revocation is only useful in combination with the ability to *detect* compromised devices, e.g., by recognizing a published set of device keys, or by examining pirated content for the presence of forensic marks.

AACS supplemental documents[2] define a simple forensic marking scheme based on the idea of "movie variants". Under this scheme, the bulk of an AACS title is encoded and encrypted normally. Content producers enable tracing by selecting a number of short segments of the content (approximately 1 second each), and generating multiple *variants* for each portion. Based on the specific keys provisioned to it, a given player will be able to decode one variant from each of these segments. To a viewer, the result is indistinguishable from a normal film. However, a tracing authority can recover the identity of the player used to decode the title by identifying subtle differences between variants. Similarly, an adversary who publishes title decryption keys can be traced via the unique decryption keys required to decrypt the movie variants.

The AACS tracing scheme suffers from a number of weaknesses. First, under the parameters suggested by AACS, successful tracing is unlikely unless the adversary has compromised 4-6 titles using the same player. Given the low cost of consumer electronic equipment, a knowledgeable adversary may circumvent the scheme at a reasonable cost, simply by purchasing new equipment on a periodic basis. Second, as the content tracing scheme relies on a small number of short variant segments, adversaries who obtain the location of these segment may choose to simply omit them from the film output. Therefore, the location of variants in a given title must remain highly confidential. However, an adversary with a compromised AACS device can easily learn the location of these areas. While the omission of variant areas reduces the out-

[2] AACS Update to HD-DVD Format Group, January 28, 2005.

put quality of the film, the absence of 10-30 seconds of material—spread out across a feature length film—may not significantly decrease its value.[3] A final concern with the AACS marking scheme is the possibility that an adversary may combine keys from several compromised devices. In this instance, he must compromise as many as six titles per compromised player before AACS tracing can successfully identify the players involved.

Compromise of Media/Title keys. Similarly, AACS does not provide an effective response against an adversary who obtains volume or title-specific keying information from a title, perhaps by compromising an authorized player. As title keys are relatively compact, distribution of keys is significantly less costly than distribution of decoded content. Furthermore, the availability of keys enables a variety of potential applications, such as PC-based ripping tools that decode content directly to disk once provided with title keys.[4] The AACS tracing scheme proposes that content creators distribute multiple sets of *sequence keys* with each title, which would help to identify the player used to recover keys. However, in the scheme proposed by AACS, content creators must detect several compromised titles in order to trace a player. Furthermore, the use of multiple key sets inevitably requires some increase in the quantity of encrypted content stored on a volume (each key variation must apply to a different segment of encrypted content.)

Specific Concerns

In addition to the above concerns, the AACS specification contains a number of specific areas that are unclear or incorrect. It is our recommendation that these issues be addressed prior to implementation of the specification.

Improper distribution of secret keys in Random Number Generation. Section 2.2 defines an algorithm for pseudo-random number generation based on the ANSI X9.31 standard. ANSI specifications note that the key k must be kept secret. While the AACS specification reiterates the importance of securing k, it also notes that "the fixed value k need not be unique per licensed product". We do not believe that secrecy of k can be guaranteed if many devices share the same k value, as this information may become public due to device compromise. In that case, it might be possible

[3] Increasing the number and size of the variation segments significantly increases the amount of material stored on disk. AACS notes that the under some example parameters (30 seconds of variant material) media storage requirements can increase by as much as 5-10% due to the additional variant material.

[4] Recovering raw (encrypted) content from AACS drives may require a firmware modification or "hack". Existing examples of such hacks include multi-region firmware hacks available for current DVD players. Note that firmware hacks designed only to recover raw data do not require compromising a drive's protected AACS subsystem.

for an adversary to compute previous/subsequent outputs of a generator based on output values, which has critical security consequences if a generator is used to compute nonces or secret keys for encryption. To address this concern, we recommend that *k* be unique for every device, in all circumstances. If this cannot be achieved, we recommend the use of a different PRNG construction that does not depend on the secrecy of the key value.

No mechanism exists for revoking recordable media. The Content Revocation mechanism allows AACS-LA to disable playback of specific titles by transmitting a revocation list to be stored at the device. However, the Content Revocation mechanism is based on Content Certificates, which are present only in pre-recorded content. The Content Revocation mechanism does not appear to contain a provision for revoking content marked as "recordable". We recommend that AACS define a Content Revocation mechanism that can be used to disable any form of playable content.

Forgery of Usage Rules. The Usage Rules associated with a title determine the set of authorized actions that a player may perform. To prevent tampering or forgery of usage rules, the rules are authenticated using a Message Authentication Code (MAC) under the title key. Because a MAC is a symmetric function, an adversary who obtains the title key (by compromising a device) can author an arbitrary set of usage rules and compute a corresponding MAC under the title key. The resulting Rules/MAC pair will authenticate as a valid rules set, and can be used to author a modified title. We recommend that this issue be addressed by replacing the MAC function with a digital signature, using an alternate mechanism to deliver the public key.

Insufficient response information in Host-Drive protocol. Section 4 of the Common Cryptographic Elements book defines a set of protocols by which a PC requests information from an AACS compliant drive. Subsections 4.1-4.4 define four protocols for querying a drive on (respectively) the 128-bit Volume Identifier, Serial Number, Media Identifier, or Binding Nonce. In each protocol, the drive response is bound to a nonce for freshness, and authenticated with a digital signature to prevent forgery or replay attacks. However, the signed response does not include the request type, so it is possible for an adversary who controls the communications channel to modify the request as it is transmitted to the drive, e.g., changing a request for a Volume Identifier into a request for a Binding Nonce. The PC will verify the digital signature on the response, and will believe that it has received a valid response to the request. To prevent such data modification, we recommend that AACS incorporate the request type into the signed response.

Use of default Initialization Vector in AES-CBC. Section 2.1.2 describes a default Initialization Vector (IV) for use in CBC-mode encryption and decryption. Unless otherwise specified, this fixed system-wide value is used at the beginning of all encryption and decryption chains. The use of a fixed IV over multiple separate encryptions is improper cryptographic practice, and can result in security failures. We recommend the use of a unique IV for each encryption chain.

Integrating Self-Protecting Digital Content with AACS

The Self-Protecting Digital Content Interface Specification (v0.95a) interfaces with the AACS subsystem through an Event that is invoked during key derivation, and a Trap that provides access to information about AACS device keys (but does not provide access to key values). These mechanisms allow content creators to hook into the AACS process and use AACS derived information in the computation of content code based transforms including forensic marking. This functionality is fully described in the Appendix to the SPDC Interface Specification.

System Overview

SPDC consists of two separate components, a Virtual Machine which executes content code, and an Interface Specification which provides an API to interface with the underlying device. The SPDC Virtual Machine provides a standard environment allowing content code to execute across various underlying architectures.

SPDC is considered to be one layer of security in a multi-layered content protection system, and complements device key management systems such as AACS. SPDC enables renewability through custom playback processing. This helps to protect content even from an adversary with the ability to compromise an authorized player.

To achieve the stated goals, the SPDC system provides several properties:

1. A standard execution environment to enable cross-device compatibility.
2. A flexible set of operations, to provide for re-implementation of arbitrary player logic as needed.
3. Robust mechanisms to facilitate code obfuscation, in order to resist efforts to reverse-engineer and disassemble content code.
4. Device resilience against the insertion of malicious or malformed content code.
5. The ability to fingerprint devices based on device-specific keys as well as unpredictable device-specific behavior, and thus vary content code execution.

SPDC achieves these goals through the use of a flexible Virtual Machine with a full instruction set. This VM is defined in the specification and can be easily implemented in a variety of hardware and software devices. Memory protection in the machine is provided using standard sandboxing techniques, to prevent malformed code from accessing data or processes outside of the machine. To protect content code , the VM instruction set provides several data-dependent instructions that make static analysis and disassembly nearly impossible, most notably the Instruction Filter. Moreover, when such instructions are passed non-simulatable values (i.e. values based on keys held only on legitimate devices) even dynamic control and data flow analyses become at best an incremental process.

While the SPDC virtual machine is extremely simple, the interface between the VM and device is somewhat more complicated. However, most of the interface can be omitted from the trusted computing base. In particular, any trap that could be emu-

lated by either the VM itself (e.g., TRAP_Memcpy) or by an external emulator without access to any device keys (e.g., TRAP_Finished) does not need to be part of the Trusted Computing Base. In fact, we believe the areas of the specification where implementations are most likely to introduce security errors are those left undefined for vendor specific behavior. It is important to note that such concerns are also present in security systems such as AACS, which do not provides the facilities for logic renewal offered by SPDC.

SPDC cannot mitigate every attack. For example, an attacker who discovers a method for easily extracting the keys from a legitimate device *and* has the ability to precisely emulate the legitimate device in software has successfully bypassed the protection provided by SPDC. Fortunately, so long as care is taken during the design of legitimate devices, such a scenario is extremely unlikely, as perfectly simulating all aspects of a device's operation (e.g., native CPU, timing, hardware, etc.) is extremely unlikely. Barring such a catastrophic failure, SPDC provides a useful second line of defense against partially compromised devices. For example, an attacker who can only recover device keys which are used in an emulator to extract protected content can be prevented from copying future titles by introducing SPDC content code that requires functionality present only in the legitimate device and not the attacker's emulator.

The SPDC Virtual Machine

The SPDC Virtual Machine specification defines a MIPS-like instruction set consisting of 59 standard machine operations (along with several reserved and vendor-defined operations.) Each machine instruction is encoded as a 32-bit value. The Virtual Machine provides content code with two memory areas, one for the content code and data, and another undefined area which can be used as defined by the device manufacturer. The VM also defines a set of 32-bit registers, a Program Counter, and an Instruction Filter, which is applied to instructions before execution.

Memory Protection. The SPDC Virtual Machine applies memory protection by masking memory access addresses to prevent them from falling outside of the designated memory areas defined by the Virtual Machine. In software, this may be achieved through a single AND operation, or in hardware by forcing certain address lines to zero. Specific device classes may define additional memory areas for access by content code. These areas are not required for general operations, and may be used to provide media buffers to content code. As noted in the SPDC specifications, device implementations should protect these memory areas, in order to prevent code from illegally accessing device RAM. This protection can be trivially achieved using a masking approach similar to the one used to protect the primary VM buffer.

Implementation-Specific Behavior. Content Code can leverage differences in SPDC implementations as a means to fingerprint devices, and to implement device-specific behavior (e.g., deploying additional countermeasures when running on potentially compromised hardware). The Virtual Machine specification enables this functionality by allowing a degree of device class and vendor-specific behavior. One instruction (VEND) is specifically devoted to vendor-defined operations, and may produce unique results on different devices. Other areas of implementation-specific device behavior include the value of register #0; the result of accessing undefined memory areas; and the execution time of player operations and traps.[5]

Instruction Filter. The Instruction Filter provides a mechanism to protect content code from static disassembly, or to enable device-specific behavior. Prior to executing an instruction, the VM first computes the XOR of the instruction with the current 32-bit Instruction Filter value. Content code may set the value of the Instruction Filter dynamically via the INSTF instruction. The Instruction filter potentially enhances a content creator's ability to obfuscate code by forcing adversaries to trace the full code execution path rather than disassembling code in a single pass. Instruction filter values may be computed via cryptographic operations, or from device-specific operations, which may require the adversary to correctly emulate this behavior in order to disassemble code.

The SPDC Interface Specification

SPDC provides content code with 26 system calls or *Traps*, 21 of which access device resources such as media, device keys or state. The interface also defines 9 callbacks (*Events*), which devices may invoke to notify content code of outside events, or to request specific actions.

Events. Players may notify content code of a variety of events, including pending or successful media reads/writes, shutdown or media eject events, player security operations and idle events sent with each frame displayed. Code may choose to ignore any of these events, or perform some custom processing. Event data is provided to content code in the Event Parameter area, which is a section of memory designated by content code. Similarly, response values are returned to the device via this same area. Code is constrained by the requirements of real-time operation, which demands that event operations return within a limited timeframe.

Device Discovery. The Interface specification defines a device Discovery trap that code may use to recover information about the device and environment on which it is

[5] While the VM Specification places bounds on the execution time of specific VM instructions and traps, the actual execution time may vary from device to device.

running. Discovery returns various information about the device, including details of attached components (e.g., devices connected to the digital output ports). This provides content creators with the means to limit playback quality, or to stop playback on detection of an unauthorized (or compromised) output component. Discovery can also return device-identifying information for use in forensic marking, including manufacturer info, device serial number, firmware version, device driver hash and RAM location, and device public key.

An additional trap (TRAP_DiscoveryRAM) provides content code with access to specific areas of player RAM external to the VM memory area. This enables code to fingerprint device software, and can be used to detect certain forms of player compromise. Device manufacturers must implement this feature with care, to prevent unauthorized access to secure memory areas containing secret information, e.g., device keys.

Slots. Compatible devices may offer content code a means to store long-term data in non-volatile RAM within the device. Each device can store data into a number of "slots". Access to slots may be restricted to code included with a single title, or shared across multiple titles. Slots can be used to store device or title-specific information, which can be used for a variety of purposes. Slots can also be used to implement Title Revocation, by recording lists of known compromised titles.

To protect slots from by unauthorized usage by content code, the interface defines a security mechanism for controlling slot access, which allows the creator of any slot to bind the slot to a particular segment of content code. This guarantees that the code operations that execute subsequent to the SlotAttach trap implement a set of known operations chosen by the slot creator. Therefore, the slot creator can limit the set of operations which execute subsequent to any successful attach operation, and can use this control to prevent unauthorized behavior.

Other Device or Class-specific Traps. In addition to Discovery, the Interface provides content code with a variety of routines that code may use to control the underlying device. This includes (but is not limited to) traps that initiate media reads; establish Internet connectivity; run native code; perform decryption with device or title keys; and generate device-signatures.

General Traps. In addition to the device-specific traps, the SPDC interface also provides a variety of traps for performance reasons. These traps provide operations that could be implemented by content code alone, but offer significantly improved performance when executed by the underlying device. These traps must be used care-

fully by implementers, but do not require any special security considerations.

Contribution of SPDC

In this section, we consider the range of threats to protected content, and discuss the assumptions we make as well as those implicit in the SPDC model. We then compare the protection offered by AACS alone against the protection that SPDC offers when combined with a key management system such as AACS.

Exceptions

Limited protection. In evaluating the SPDC system, we make several stipulations. First, neither SPDC nor any client-side rights management system can provide unlimited protection against a sufficiently motivated adversary. If an adversary has the means to play content using a legitimate, non-revoked player, he has the means to eventually defeat most security measures that prevent unauthorized usage. Furthermore, SPDC code must potentially protect content even while running on a compromised device. Consequently, it is impossible to guarantee that a particular piece of content code will survive a prolonged attack designed to circumvent the security function and allow unauthorized access to content. The primary goal of the SPDC framework is to offer renewability, by allowing content producers to author customized content code for each title. This forces adversaries to repeatedly invest resources against individual titles, rather than achieving a full system compromise in a single effort.

Media Duplication. Both AACS and SPDC provide mechanisms to revoke individual titles in the event of illegal media replication. SPDC can achieve this by storing identifying information for compromised titles into non-volatile storage (slots), which content code can access prior to decoding a title. Because the details of the AACS title revocation mechanism are not publicly available, we do not include this functionality in our comparison.

Recovery of traceable content. There is very little that any content system can do to prevent the recovery of some output from a legitimate device, although such output may be traceable. Specific devices may limit the quality of this recording by limiting access to high-quality digital outputs, or by employing mechanisms to degrade content quality in the presence of a compromised device; however, evaluating such mechanisms requires detailed consideration that is outside the scope of this comparison. We do not compare AACS and SPDC in their response to this attack, and simply note that both AACS and SPDC can provide for forensic marking, playback restrictions, and other mechanisms (e.g., online registration).

Attack Types

In order to evaluate the range of threats to content security, we first define a range of attacker goals and attack scenarios. Table 1 contains an overview of the performance of AACS versus a combination of AACS and SPDC in each scenario, which we describe in detail below.

	Output Only (No Compromise)		Single-Device Compromise		Device-class Compromise	
	AACS	SPDC+ AACS	AACS	SPDC+ AACS	AACS	SPDC+ AACS
1. Unauthorized Access	●	●		●		●
2. Creation of Untraceable Content	●	●		●		●
3. Title-key Compromise	●	●		●		●
4. Class or System Compromise	●	●	●	●	☼	●

● = Attack addressed,
☼ = Attack partially or incompletely addressed,
(no symbol indicates attack not addressed, or addressed only for future titles)

Table 1: Security properties of AACS alone vs. combination of SPDC/AACS, considered under various attack scenarios. We consider the immediate protection offered for a given title, whereas AACS revocation provides protection only for *future* titles. Furthermore, AACS requires device-class revocation to recover system security in the case of device-class compromise.

Unauthorized access. In addition to preventing duplication, a protection system should provide for limitations on content usage, e.g., prohibiting certain actions or limiting playback quality. An attacker may attempt to circumvent these restrictions and use content in an unauthorized manner.

Creation of untraceable content. A likely attacker goal is to recover high-quality *untraceable* content from a device. This requires the adversary to defeat or bypass any marking subsystem included in the device or content code.

Title-key compromise. An adversary may seek to facilitate playback of a given title on unauthorized devices, e.g., a software device emulator running on a personal computer. This requires that the adversary obtain title or media keys, and either bypass

content code or emulate a device such that content code operates as it would on an authorized device.

Class or system compromise. An adversary may seek to permanently compromise an entire content protection system by developing an easily-replicated compromise that *cannot* be fully avoided even through the renewability features of the system. This is by far the most powerful attack on such a system, and implies that the adversary has overcome all security measures and can compromise any title at will.

Compromise Scenarios

Content code may be faced with a variety of adversaries with differing levels of skill and resources. We assume that all adversaries have access to one or more devices, the public SPDC/AACS documentation, and the skills to implement software tools. In this section we consider three scenarios that differ primarily in the adversary's degree of access to protected device internals such as keys and VM implementation. The scenarios we consider are:

Output only (no compromise). This is the ideal case for a content protection system, in which the adversary has no access to compromised device internals. All critical components of the protection system are secured by protected hardware, and the adversary cannot access secret values (e.g., device keys) or prevent correct system operation.

Single-device compromise. With sufficient effort and resources, an adversary may circumvent the hardware protection on a single device (or small number of devices). This potentially grants the adversary full access to all keys and secret information stored on the compromised device, and allows him to tamper with the device at will. However, this hardware compromise is not easily replicable; each device compromise requires an expensive and time-consuming effort for the adversary.

Device-class compromise. This attacker is able to easily bypass the hardware protection of any device in a given class, granting him ready access to the secrets stored on a large number of devices. This adversary can potentially repeat this compromise whenever necessary to overcome AACS revocation (e.g., by purchasing a new player to replace one that has been revoked).

Table 1 above summarizes the protection offered under a variety of attack scenarios. The feasibility of each attack is considered when only the AACS key management system is in place, and again when the SPDC system is implemented on top of AACS. Each of these attack scenarios is discussed in detail below.

Device Output Only (No Compromised Devices)

In the case where adversaries are unable to compromise devices, AACS and SPDC/AACS provide similar degrees of content protection. Adversaries are prevented from using the provided media in a manner that is prohibited by AACS policy, and protected hardware may perform device-specific marking to facilitate device revocation.

Under all circumstances, the adversary can obtain a copy of the content that contains some form of detectable mark, e.g., by re-encoding the analog output signal. The new form can then be repeatedly copied and distributed physically or via the Internet. In this case, both AACS and SPDC/AACS provide the means to limit access to high-quality digital outputs depending on policies. Assuming robust marks, content producers can detect and revoke devices used to produce pirated content. Since protected hardware secures the marking and decryption processes, recovering untraceable content, or any key material (to facilitate title or system attacks) should be difficult or impossible in this scenario.

Single device compromise

In a system implementing AACS alone, a user with a compromised device is potentially capable of recovering and distributing title keys, media keys and device keys specific to the player. Given access to any of these keys, a user can easily decode digital content on a home PC, following the AES decryption procedures defined in the AACS specification. In this circumstance, AACS does not offer protection for the compromised title, but provides a means to protect future titles. Upon detecting the unauthorized use or distribution of device keys, content producers can revoke the device in future titles—although this will not protect any titles issued prior to revocation. Assuming that device compromise is rare, revocation allows producers to author new titles that address an occasional isolated device compromise.

The addition of SPDC content code significantly increases the investment required to compromise a given title. With SPDC, an adversary who compromises a player cannot distribute title-specific keys for use outside of the player, without also reverse-engineering the SPDC content code, or providing a software emulator that perfectly emulates an authorized device. Even when the adversary is successful in this costly effort, his benefit is limited to the title compromised. Note that SPDC provides this protection immediately, even if content creators are unable to detect and revoke the compromised device. Because Content producers can issue new content code in each new title, adversaries are forced to invest significant additional time and effort to compromise each title release.

Device-class compromise

The additional layer of security provided by SPDC is particularly important in the case where an entire device-class has been compromised. In this scenario, an adversary can compromise new devices at will, replacing each as soon as it is revoked by AACS. If an adversary can replace each revoked device at a reasonable cost, the benefits of revocation are significantly reduced. To prevent further title compromise, content producers may have no choice but to revoke the entire class of devices. This action may be extremely costly and inconvenient to legitimate users.

Some have argued that applying strict "robustness rules" to device implementations can prevent device-class compromise. Such rules attempt to enforce hardware protection by committing manufacturers to a series of implementation guidelines and security mandates. Given the system-wide consequences of a security failure in a single player model, we believe that this approach is dangerously insufficient. Even when implemented by a single experienced manufacturer, security systems are prone to some incidence of failure. Attempting to guarantee secure implementation across a variety of manufacturers substantially amplifies this risk.

In the case of a device-class compromise, SPDC provides several additional forms of protection. As in the single-device compromise scenario, the adversary must still reverse-engineer each new piece of content code shipped with a new title in order to overcome the protections offered by SPDC. Furthermore, with knowledge of the compromise, content producers can author code to detect the presence of a compromised device, and take various actions to further mitigate compromise, e.g., protect title keys, use alternate content, apply firmware patches, reduce content quality, etc. Additionally, depending on the characteristics of the device and the nature of the compromise, it may be possible for content producers to develop specific tests that could detect an ongoing compromise, and subsequently abort content processing.

Conclusion

This report examined the Self-Protecting Digital Content system, and compared the security offered by SPDC used in conjunction with AACS to the security offered by AACS alone. We conclude that the protection offered by AACS alone is insufficient to protect content in the event of various device compromise scenarios, and that SPDC can provide an additional level of security that is sufficient to withstand many likely attacks against a content protection system.

No device-side content protection system can withstand all attacks. We note explicitly that the protection offered by any piece of SPDC content code should be considered time-limited. The principal advantage of SPDC is the ability to renew system security, potentially with each new title released. SPDC also provides mechanisms by which content creators can overcome device implementation flaws, and provides a generally more resilient level of system security than can be provided by key management systems alone.