



**GiantSteps**  
Media Technology Strategies

250 West 57<sup>th</sup> Street, Suite 1020  
New York NY 10107  
212 956 1045  
fax: 212 581 1352  
[www.giantstepsmts.com](http://www.giantstepsmts.com)

---

# The New Technologies for Pay TV Content Security

---

By Bill Rosenblatt

August 19, 2011

**irdeto**

---

## Table of Contents

Table of Contents.....	2
Introduction and Executive Summary.....	3
Background and History .....	4
Security Challenges of the Digital Economy.....	4
Evolution of Pay TV Security .....	6
Current Best Practices for Content Security .....	13
Generic Technology Components .....	13
Client Security Techniques .....	15
Beyond the Server and Client: Emerging Techniques .....	22
Conclusion.....	27
About Irdeto .....	27
About the Author .....	28
About GiantSteps Media Technology Strategies .....	28

## Introduction and Executive Summary

Pay television often leads the way in content protection technologies. That's because among digital content delivery modalities, pay TV is unique for two reasons: one technical, one economical.

The technical reason is that many digital pay-TV systems provide some kind of communication channel from the client device, such as a set-top box, back to the server at the head end. That means that the device is able to "phone home" for various purposes, such as to register itself on the network or respond to various types of security messages. Such two-way communication capability can facilitate some of the advanced security techniques discussed here that would otherwise be impossible to implement. Contrast this with delivery modalities such as physical media (DVDs, Blu-ray discs), where no server connectivity can be assumed at all, or even Internet delivery modalities such as PC downloads, where consumers expect to be able to use content offline.

The economic reason for pay TV's uniqueness is that the incentives of pay-TV operators (such as cable and satellite providers) and content owners (such as movie studios) regarding content protection are aligned. The content owner doesn't want its copyrights infringed, and the operator doesn't want its signal stolen. Again, contrast this with other delivery modalities: makers of consumer electronics (such as SmartPhones) generally don't build content protection technologies into their devices, and content retailers would rather not have to pay the cost of implementing DRM and similar technologies. As we'll see in this white paper, some of the more notable failures in content protection technology have been due to consumer device makers trading security off in favor of lower unit cost.

For these reasons, pay TV has often been in the vanguard of content security technology, and it's one of the real "success story" areas of the field. A set of techniques has emerged in pay TV that has the potential to take content security to new levels of effectiveness and total cost of ownership over time. We'll describe them here.

In this white paper, we will look at the history of technologies used to secure digital pay TV, starting with the introduction of the Digital Video Broadcast (DVB) standard in 1994. We'll explore some of the weaknesses inherent in the DVB standard itself as well as successful hacking strategies such as smart card cloning and control word sharing, and the techniques used to guard against them. We will also look at the history of DRM, which largely parallels the development of DVB, and see examples of the hacks that have occurred owing to security weaknesses in those technologies.

In the second part of the paper, we will explore the set of techniques that are considered today's best practices for pay TV security. We'll examine some of the advances in security technology made possible by Moore's Law and the introduction of building blocks such as elliptic curve cryptography (ECC), flash memory, and secure silicon. And we will discuss inter-related techniques such as individualization, renewability, diversity, and whitebox cryptography. We'll introduce the concept of *security lifecycle services*, which tie many of these techniques together into end-to-end schemes for security and threat monitoring over time.

In the final section of the paper, we will look at content identification techniques designed to find unauthorized content "in the wild" (on the Internet) and trace it to the users or devices that made it available. These techniques are complementary to the pay-TV security techniques discussed in the remainder of the paper.

## Background and History

In this section, we will take a trip through the history of security techniques for digital broadcast video. We'll look at the challenges to content security that have arisen from the proliferation of devices and services. Then we'll see how evolutions in memory, networking, and cryptography technologies led to improvements in broadcast video security that addressed weaknesses exploited by hackers. Understanding of how security techniques evolved should provide a basis for understanding the need for and value of the new techniques presented in the following section.

### Security Challenges of the Digital Economy

As content has gone digital, movie studios, television networks, and other content owners have lost more and more control over their intellectual property. At the same time, the consumer electronics industry has introduced generation after generation of devices that make content available in more places, and more cheaply, than before.

#### Devices: The Second Dimension of Content Security

Content providers have to make their content available on an increasing variety of devices; yet those devices need to have enough security to make content owners comfortable in licensing their content for use on those devices. For pay television, the responsibility for content security lies primarily in the hands of operators, such as cable and satellite service providers. In most cases, operators influence consumers' choices of reception devices, either by purchasing them on subscribers' behalf or by approving them for use on their networks. This means that device makers also end up sharing in the responsibility for content security more directly than they do for other content delivery modalities, such as Internet downloads or "over the top" services.

Even so, operators face the biggest set of security challenges of any players in the content value chain. Until recently, the challenge could be characterized in the stereotypical way: as a "race against hackers." Operators would deploy a certain security technology; hackers would figure out how to circumvent it; operators would respond with a technology that plugs the hole; the cycle would repeat.

But more recently, the challenge for operators has taken on a second dimension: the proliferation of devices that can consume broadband video beyond set-top boxes (STBs), and Hollywood studios' interests in making their content available on these new devices. Consumers are no longer forced to watch certain content on their STBs; they can and should be able to watch it on IPTVs, tablets, handsets, and so on.

*The race against hackers has become a simultaneous race to keep up with device proliferation.*

---

Operators need to be able to offer premium content on the entire range of desirable devices while adhering to the studios' security requirements. Thus the race against hackers has become a simultaneous race to keep up with device proliferation. This means, among other things, working with device makers to deploy security schemes that are as interoperable across device platforms as possible. Ideally, security technologies can be instantiated for multiple platforms automatically and scalably.

Security technology for pay TV and broadband video has evolved significantly since the advent of DVB in 1994. Before we discuss this evolution, it should be helpful to discuss the state of cryptography at that time – and therefore the kinds of hacks that operators needed to prevent.

## Crypto Key Protection

At the heart of just about every digital content protection scheme is a cryptographic algorithm or cipher. When pay television went digital in the mid-1990s, crypto algorithms were strong enough to make cryptanalysis – trying to crack the algorithms themselves – far too difficult to be worth the effort. Therefore, hacking schemes for digital content worked “around” the algorithms themselves and focused on other attacks, such as discovering the keys used in the algorithms.

*Hacking schemes for digital content work “around” cryptographic algorithms and focus on other attacks, such as discovering the keys used in the algorithms.*

---

At that time, there were two general types of crypto algorithms: *symmetric-key* and *asymmetric-key*, a/k/a public-key or public/private-key. In symmetric algorithms, the same key is used to encrypt and decrypt the content. The key must be sent from a server to a client. It must be protected both in transit and on the client device, lest it be discovered.

The U.S. government standard symmetric crypto algorithm was DES (Digital Encryption Standard), which dated back to the 1960s. By the mid-90s, the standard practice was to apply DES three times with different keys (resulting in Triple DES or 3DES) because then-current computing power rendered the original DES too weak. Advanced Encryption Standard (AES) replaced DES as the U.S. government standard in 2001 after a competition won by a team of Belgians and their “Rijndael” algorithm<sup>1</sup>. The first important public-key algorithm was the RSA algorithm, which Ronald Rivest, Adi Shamir, and Leonard Adleman of MIT invented in 1978.

Public-key algorithms had the advantage of isolating private keys on client devices (such as set-top boxes), meaning that if someone were to hack a server and steal all of the keys stored there, they would be worthless, because it is too difficult to discover private keys from public keys. However, asymmetric algorithms had two disadvantages to being used for this application: they were far too inefficient to be used with large amounts of data such as digital video content, and they required more complex distributed key management schemes.

Therefore most content protection schemes used symmetric-key algorithms. This meant that the task of hacking these systems became the task of discovering the secret key. An important step towards key discovery was reverse engineering the client-side code that obtained the key and used it to decrypt content, so that the hacker could figure out where and how the key was stored.

Thus the two main tasks of content protection schemes were preventing keys from being discovered and preventing code that handles keys and content from being reverse engineered.

---

<sup>1</sup> J. Daemen and V. Rijmen, *The Design of Rijndael: AES — The Advanced Encryption Standard*, Springer-Verlag, 2001.

It was axiomatic in the mid-90s that the only effective way to protect keys and key-management code was to store them in tamper-proof hardware, instead of storing them in insecure client device memory that could easily be examined by hackers. This gave rise to the *smart card*, which contained EEPROM (Electrically Erasable Programmable Read-Only Memory) that stored keys. The smart card traces its origins to the French company Honeywell Bull in the late 1970s; smart cards were originally used in credit cards and other applications.

## Evolution of Pay TV Security

This section recounts the major steps in the evolution of digital pay TV security, which are shown in Figure 1. This story focuses entirely on operator-controlled devices, i.e., STBs.

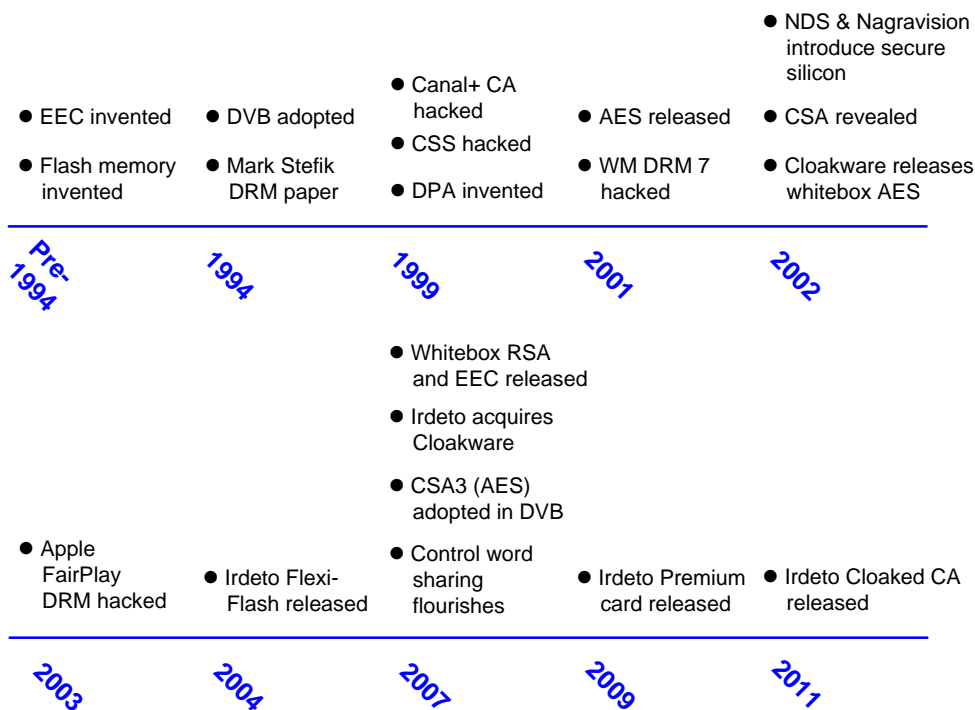


Figure 1: Timeline of pay TV-related security developments.

## DVB Origins and Security Limitations

Security technology for digital pay television evolved out of a combination of digital smart card technology and conditional access techniques used in analog TV. The watershed event in digital pay TV development in many parts of the world was the establishment of the Digital Video Broadcasting (DVB) set of standards by the DVB Project<sup>2</sup>. DVB-C and

<sup>2</sup> [www.dvb.org](http://www.dvb.org).

DVB-S, for cable and satellite respectively, were established in 1994. (DVB-T for terrestrial broadcasting and DVB-H for handheld devices came later, in 1997 and 2004 respectively.)

The first generation of DVB conditional access smart cards, as shown in Figure 2, included a few kilobytes of memory. Content was scrambled using the Common Scrambling Algorithm (CSA), which was intended to be implemented in hardware. Enough of the details of CSA were initially kept secret so as to prevent reverse engineering. The algorithm was discovered in its entirety after the release of a software implementation in 2002. In 2007, the DVB Project replaced CSA with CSA3, which uses 128-bit AES, although this is not in wide use for backwards compatibility reasons.

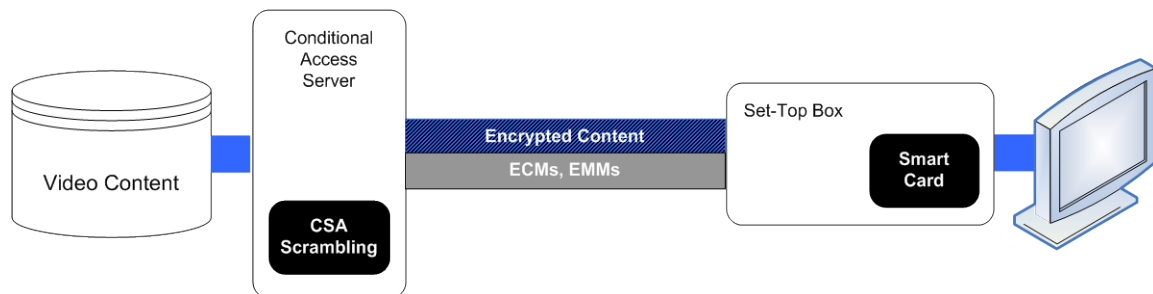


Figure 2: First-generation DVB conditional access architecture.

The DVB conditional access standard specified a CSA content key, called a *control word*. *Entitlement control messages* (ECMs) sent to set-top boxes contained control words.

Despite the fact that smart cards put both keys and key management in hardware, making them very difficult to reverse engineer, first-generation DVB conditional access schemes suffered from various security weaknesses, including:

- **Control word protection:** The standard itself did not specify how control words could be kept secure in transmission or in residence on the client device. Some vendors developed proprietary schemes for protecting control words en route to the client device, while others did not.
- **Inefficient renewability:** In many implementations, the security scheme could not be changed without swapping out smart cards, although the DVB standard did not preclude over-the-air renewability.
- **Key management visibility:** Processes for key management were not obligated to be obfuscated or firewalled, thus enabling reverse engineering in implementations that did not do this.
- **Susceptibility to blocking and replay attacks:** CA systems whose control messages contained instructions to turn users' subscriptions on and off could be fooled if such messages were blocked or replayed.

*The DVB standard did not specify how control words (keys) could be kept secure in transmission to client devices, so some vendors developed proprietary schemes for protecting them.*

---

Hackers had a number of ways to break this system. They could clone valid smart cards at factories or by re-activating smart cards that had been deactivated (e.g., because a customer cancelled a subscription). There is also a technique called differential power analysis (DPA), invented by Cryptographic Research Inc. (CRI)<sup>3</sup>, in which a hacker could analyze a smart card's power consumption in very fine detail as it executes a cryptographic algorithm in order to determine how it works and eventually deduce the key. However, this technique is mostly of theoretical interest and not known to be widely used.

The only remedy available to hacks to first-generation DVB smart cards was *key cycling* through *entitlement management messages* (EMMs). EMMs were used in DVB conditional access to deliver keys to smart cards; they were sent at regular intervals. New keys could be delivered in order to enable new services for subscribers or to update security. Not all vendors used the key-cycling technique.

To make the system more secure, EMMs could be sent more frequently, or if a hack was detected, an EMM could be sent specially to force keys to be updated. However, this was merely a temporary fix, as the process of key discovery remained the same; hackers merely had to use the same technique to discover the new keys.

By 2000, it had become clear that the security architecture for DVB was inherently incapable of surviving large-scale security threats. The first response to this realization was some automation of key-cycling so that it could happen more frequently and without intervention by operators or security vendors.

A more substantive response was to accelerate the pace of developing and rolling out new smart card designs. After a few years, the installed base would include several varieties of smart cards. This added to support burdens but increased security because techniques for hacking one type of smart card would not necessarily apply to other types.

In addition, the physical security of the cards themselves had also been enhanced to incorporate temperature, light and electrical stimulus detectors to guard against hardware-based hacks.

#### The Influence of DRM

Around the same time, digital rights management (DRM) techniques for content distributed over the Internet began to appear in earnest. DRM technologies were first patented and commercialized in the mid-to-late 1990s, but they originally applied to publishing (text documents) and music. DRM for video was understood on a technical level but not widely used because of Internet bandwidth limitations for most users.

A generic DRM architecture is shown in Figure 3. DRM was originally intended for downloaded content instead of streaming. In DRM systems, encrypted files are sent from a content server to a client device (or software). Many DRMs use small encrypted files called *licenses*, which are roughly analogous to EMMs in DVB: they contain encryption keys, identification of the device and/or user for which the content should be privileged,

---

<sup>3</sup> P. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis*, pp. 388-397, *Advances in Cryptology — CRYPTO '99* (LNCS 1666), Springer-Verlag, 1999.

and sometimes a precise indication of the rights that the system is conveying (e.g. play for a month, play 10 times, etc.).

*Virtually all DRM implementations have been entirely in software, leading to the development software-based techniques for hiding encryption keys.*

Some of the techniques of DRM began to creep into the world of conditional access for pay television. Seminal work on DRM technology from EPR (later Intertrust) and Xerox PARC envisioned hardware-based implementations, and in fact warned of the inherent weaknesses of implementing content security on a platform such as the PC<sup>4</sup>. Yet virtually all DRM implementations were entirely in software, meaning that software-based techniques for hiding encryption keys had to be developed.

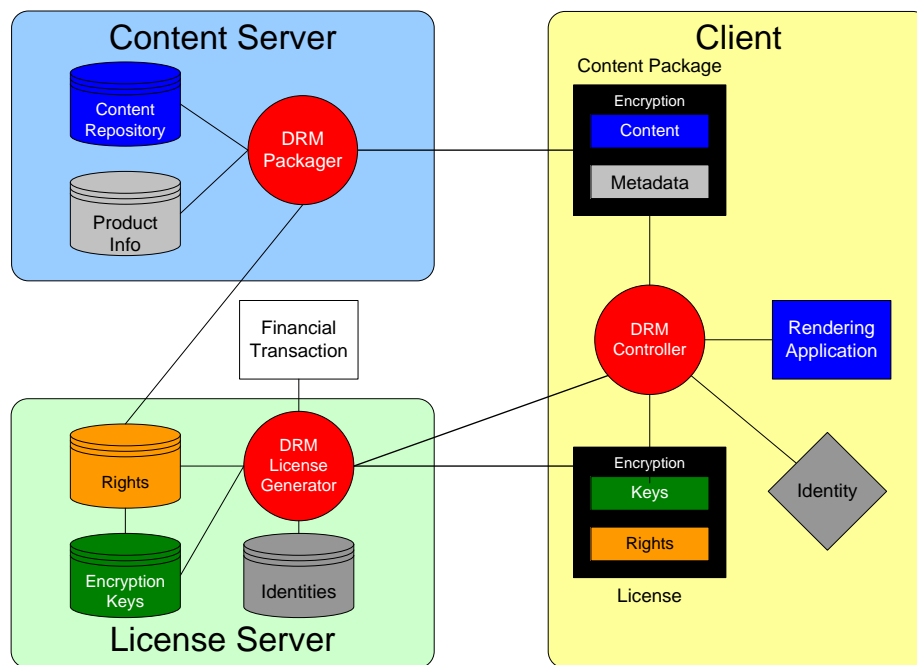


Figure 3: Generic architecture of DRM systems. Adapted from Rosenblatt, B. et al, *Digital Rights Management: Business and Technology*. Hoboken: John Wiley and Sons, 2001.

Both DRM and conditional access center on encryption of content. But DRM techniques<sup>5</sup> had important differences from conditional access, including these:

- As mentioned above, DRM was originally intended for downloads. This meant that the content would not just play but could continue to reside on the client

<sup>4</sup> See for example Mark Stefik of PARC's classic 1994 paper *Letting Loose the Light: Igniting Commerce in Electronic Publication*. Available as a chapter in M. Stefik, (Editor). *Internet Dreams: Archetypes, Myths, and Metaphors* (MIT Press, 1997).

<sup>5</sup> In general; see B. Rosenblatt, *Digital Rights Management: Business and Technology* (Wiley, 2001) for a reference architecture and general explanation.

device. This led to a requirement to secure content resident on the device after download.

- DRMs can authenticate entities other than single client devices to use content. For example, they can authenticate by user ID (e.g. username and password regardless of device) or by combination of user and device ID.
- Because DRM implementations were all done in software, content distributors had greater flexibility to implement new business models on the fly. In fact, some DRM schemes support *dynamic rights*, i.e. changing license terms even after the user has already downloaded the content.

*DRM technologies were developed to meet certain requirements that crept into pay TV over the years, such as content downloads, greater business model flexibility, and authentication by user ID.*

---

### Control Word Sharing

The next significant innovation in digital conditional access was a software technique that enabled software updates to be sent to cards already in the field. Memory available onboard smart cards also increased, leading to the ability to include some usage rules in downloaded entitlement messages: for example, enabling the user to purchase a series of pay-per-view events at a discount.

Yet around the same time, another source of rampant piracy came into being: sharing of control words over the Internet. As mentioned above, the DVB standard did not encompass the protection of keys to control words in transit. Although some vendors (such as Nagravision and NDS) encrypted the keys, others did not.

Hackers could thus capture control words and post them on websites. They could use smart card emulators, which were circuit boards that mimicked smart cards and could be programmed to use stolen control words. In fact, such devices were not even deemed illegal in some countries. This practice flourished around 2007. The Internet has further facilitated hacking by enabling websites and forums where hackers can communicate their techniques and advertize their exploits.

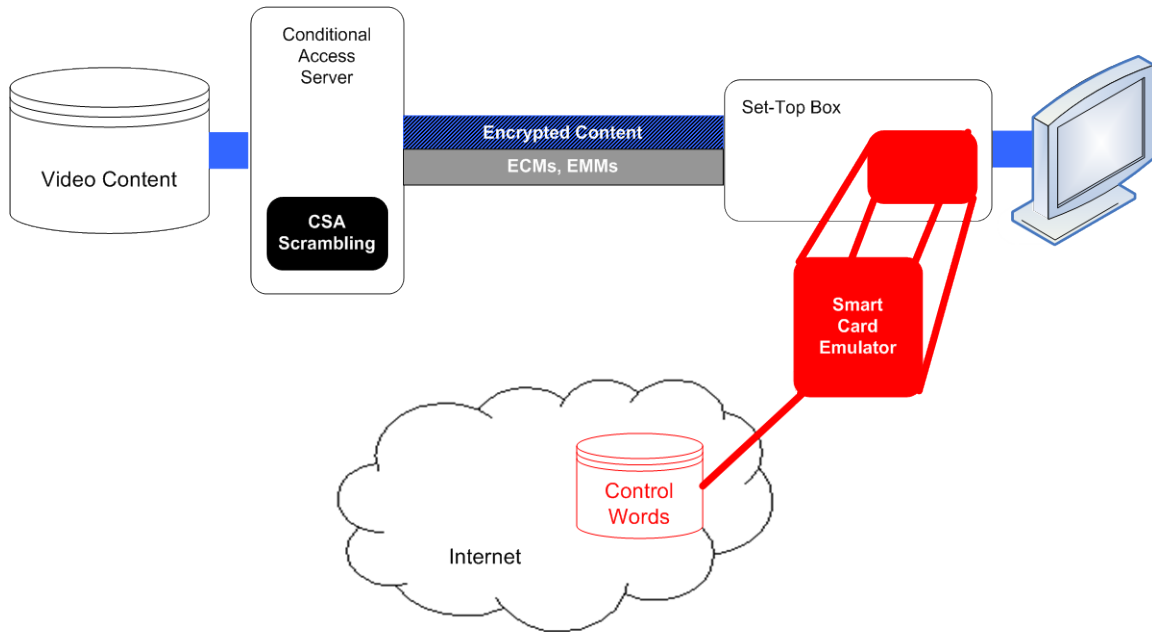


Figure 4: DVB hacking via control word sharing over the Internet.

#### Other Content Security Failures

We can also look outside the world of digital video broadcasting and find several examples of content security systems that have failed because of poor design, or due to constraints in cost or hardware capabilities. Here are a few:

*Other content security schemes have failed for reasons including software key discovery, lack of diversity, lack of renewability, and broken roots of trust.*

- **Content Scramble System (CSS) for DVDs: weak algorithm, single keys for all content.** CSS was famously hacked by Jon Lech Johansen and two other Norwegians in 1999. The strength of CSS was hampered for several reasons, including limitations on key lengths in exportable crypto algorithms, design problems that reduced the effective key length, and the desire of the consumer electronics makers who designed it (Toshiba and Matsushita, now Panasonic) to make it as cheap to implement in DVD players as possible. The result was an algorithm that could even be broken by brute force attacks on a PC in minutes. But the real Achilles' heel of CSS was that it relied on a single set of keys: once those keys were leaked, the hack could use them to unlock any DVD.
- **High Definition Copy Protection (HDCP): broken root of trust, lack of renewability.** This protocol was developed by Intel for protecting video content in transition from one device to another, such as from a Blu-ray player to a TV monitor. HDCP was hacked around September 2010<sup>6</sup>. Some cryptographic researchers have suggested that the hack was due to the particular configuration of the crypto algorithm (Blom's Scheme, from the early 1980s), which traded off

<sup>6</sup> <http://copyrightandtechnology.com/2010/09/19/assessing-the-hdcp-hack/>.

cost for security and made it possible to hack the algorithm itself. But others contend that the hack was made possible because some master keys from the root-of-trust facility were leaked or discovered. In HDCP, devices have private keys which the discovered master keys can be used to generate. HDCP also has the ability to revoke devices whose private keys have been discovered. But the keys can't be renewed. The hack made it possible to generate new device private keys at a rate so fast that revoking them became a futile exercise.

- **Microsoft Windows Media DRM (WMDRM) 7 and Advanced Access Content System (AAC3): software device key discovery, lack of renewability, lack of diversity.** Hackers found ways of discovering device keys hidden in software for WMDRM 7 and AAC3. These were done in 2001 and 2007 respectively by hackers who called themselves "Beale Screamer"<sup>7</sup> and "Muslix."<sup>8</sup> AAC3 is one of two sets of security schemes used in Blu-ray players; at the time, it was also used in the now-defunct HD DVD format. Both schemes used strong crypto algorithms but hid keys in software (in the case of AAC3, a software player for Windows PCs was hacked). In both schemes, keys could not be renewed, nor could the methods used to obfuscate them. Incidentally, both hacks had limitations that blunted their impact: the Beale Screamer hack and subsequent hacks to WMDRM required that the user legitimately obtained the content in the first place; the Muslix hack to AAC3 only exposed the device key and still required other keys, such as title keys (content keys), to work.

The challenge for content security technology designers now and in the future is thus threefold:

1. Break the cycle of hack-patch-hack-patch.
2. Minimize the impact of hacks on the overall security of content distribution systems.
3. Minimize the long-term cost of content security.

In the next section, we will describe techniques that have been developed recently with the aim of meeting these goals.

The further and broader challenge is to develop techniques that work for a wide range of devices. Some of the techniques we describe in the next section were designed for STBs, but where possible, they need to be adapted to the panoply of handsets, tablets, netbooks, and other devices on which consumer expect to use content.

---

<sup>7</sup> <http://cryptome.org/beale-sci-crypt.htm>.

<sup>8</sup> <http://freedom-to-tinker.com/blog/felten/aacs-decryption-code-released> and follow-up posts.

## Current Best Practices for Content Security

In this section, we describe content security techniques for digital pay television that are considered current best practices. These are all complementary to one another, in that they all have their places in content security systems, though they may apply to different aspects of such systems. For example, whitebox cryptography is antithetical to “black box” style code protection, but both can be used in the same system: for example, whitebox cryptography could be used to protect symmetric encryption keys, while black box techniques could be used to protect other critical code.

Some of these techniques arose out of the pay-TV/STB world, while others arose out of DRM for Internet-connected devices or out of IT security in general. Yet all of these techniques need to be drawn on as content service providers work to satisfy user demands for content on multiple platforms and proliferating content business models.

The best-practice techniques apply to content receiver devices as well as other devices that consumers will use with the same content. In fact, the ideal techniques are those whose implementations can be designed once and automated to apply across multiple platforms. Irdeto’s Cloaked Conditional Access (Cloaked CA) technology, released this year, incorporates many of the techniques described here including individualization, diversity, renewability, code obfuscation, and whitebox cryptography.

After best-practice techniques, we describe some emerging technologies that attempt to cover content security “in the wild,” outside the realms of servers and their connections to home or portable clients.

### Generic Technology Components

The first type of best practice we will cover is generic technology components that are useful as “building blocks” in client devices and software to achieve improved security.

#### Elliptic Curve Cryptography

Recently developed cryptographic techniques improve on some of the tradeoffs inherent in traditional symmetric and asymmetric-key schemes. Recall that symmetric-key algorithms are efficient to use on large amounts of data (content) but require that the server and client share a secret key, which is used both to encrypt and decrypt the data. The requirement that symmetric keys be present on both server and clients leads to security vulnerabilities. Asymmetric-key algorithms, on the other hand, reduce security vulnerabilities but are far less efficient and require much longer key lengths for adequate security.

The most promising solution to this tradeoff is *elliptic curve cryptography* (ECC). ECC was discovered independently by Victor Miller at IBM in 1985<sup>9</sup> and Neal Koblitz at the University of Washington in 1987<sup>10</sup>. ECC is a more recent form of public-key crypto that is more efficient to implement than the traditional RSA method of nine years earlier. The RSA method is based on some mathematics around the multiplication of two large prime numbers, which (as mentioned above) are inefficient to compute and requires large keys.

---

<sup>9</sup> V. Miller, Uses of elliptic curves in cryptography. In Proceedings of Advances in Cryptology (CRYPTO '85), 1985, pp. 417-426.

<sup>10</sup> N. Koblitz, Elliptic curve cryptosystems, in *Mathematics of Computation* 48, 1987, pp. 203–209.

ECC is based on a different set of mathematical concepts – computing points on elliptic curves.

ECC keys can be about the same size as AES keys (e.g., 128 to 256 bits) while providing security strength equivalent to RSA keys thousands of bits in length. The National Security Agency “endorsed” ECC by including it in its Suite B of crypto algorithms in 2005, along with AES for symmetric encryption.

*With Elliptic Curve Cryptography, it is possible to eliminate bulk key discovery on servers without introducing much inefficiency of storage and data transmission.*

---

With ECC, it is possible to eliminate server-side bulk key discovery (as is the case with RSA) without introducing much inefficiency of storage and data transmission into conditional access systems. Still, ECC (again, like RSA) is not efficient enough to use on video content. The only real test of the strength of a crypto algorithm is the test of time. ECC has been used in DRM technology since at least 2001 (Version 7 of Microsoft Windows Media DRM used 160-bit ECC) and the algorithm itself has not yet been cracked.

#### Flash Memory

Flash memory is a more recent type of EEPROM, introduced in the late 1980s, that is faster and takes up less physical space for a given amount of storage than older types of EEPROM.

The move from older-style EEPROM to flash memory in smart cards creates the possibility of a paradigm shift in digital video client security architecture: smart cards need no longer be constrained to holding keys and key management functionality. With flash memory, it becomes possible to put much more code on the smart card, where it is more secure than in regular client device RAM. In fact, it is now possible to put code for the equivalent of an entire typical DRM client onto a flash memory smart card, leading to all the flexibility that DRM provides, as discussed above, with greater resistance to tampering and reverse engineering. Digital broadcast video security providers are only now beginning to tap these possibilities.

One example of the use of flash memory in smart cards is Irdeto's Epsilon card, which first appeared in 2004 and used a flash memory scheme called FlexiFlash for secure over-the-air updates to cards. Irdeto's current flash memory smart card is the Premium card.

#### Secure Silicon

Once it became possible to put more business-rule and key-management code into secure smart card memory and to rewrite that code remotely (without necessitating service calls or “truck rolls”), designers began to look at software-based solutions from the DRM world and see what could be used from them.

Many DRM architectures are based on the concept of *root of trust*. The idea is to start with key material created in a completely protected environment (the root of trust), and then use crypto mathematics to generate additional key materials outside the protected environment (e.g., on client devices) that are difficult to discover without knowing the root of trust key(s). This increases the difficulty of discovering other keys on client devices, and therefore makes it somewhat safer to put client security code entirely in software.

In DRM systems, the root of trust is usually a server facility; for example, Intertrust Technologies has a division called Seacert, which acts as root of trust for the Marlin DRM and operates a highly secured hardware facility. Such DRMs cannot rely on roots of trust in client devices such as generic PCs. But in content distribution schemes for STBs and other types of client devices with appropriate hardware, it can be possible to rely on secure IDs built into those devices.

The most effective way to do this is to embed security keys in silicon at chip manufacturing time. This means that STB chip makers like ST Microelectronics, Broadcom, and NXP would need to create chips that all have unique keys. This is more expensive and time-consuming than traditional techniques of manufacturing large amounts of identical chips.

*The total cost of ownership of security technology for broadband operators is lower with STBs with roots of trust embedded in silicon at chip manufacturing time.*

---

However, broadband operators have come to realize that their total cost of ownership is lower when they have STBs with hardware roots of trust. Because many of them pay for STBs, they have pushed content security vendors to work with chip makers to create STB chips with individualized keys. NDS and NagraVision pioneered this development around 2002. Since then, the chip makers have figured out ways to make the individualized chips more efficiently – to the point that this process is the norm today.

The result of a client device with a hardware root of trust is the ability to implement full-scale DRM functionality on the client side with better security than software-only DRM solutions – the best of both worlds.

## Client Security Techniques

Now we look at security techniques that are inherent in content security solutions. These are used to prevent reverse engineering, software tampering, and/or crypto key discovery.

### Individualization

Individualization refers to binding client code to a specific device. Once content security code is individualized, it can only be used on that client device and cannot be implemented (“cloned”) on any others.

Individualization requires some kind of device identification. The best source of this is a unique and secure identity built into the device, such as a serial number stored in tamper-proof hardware, or secure silicon as described above. Yet many devices do not have suitable serial numbers: they aren’t secure enough, they aren’t included for cost reasons, or they aren’t included for privacy reasons – the most prolific example of the latter being PCs.

If a device does not have a suitable unique ID, then it is up to the security system to establish one as soon as possible – as part of application installation or after a generic application is downloaded into the device, and before any content is distributed to it. The best way to do this depends on the device type. If the device is one of many functionally identical devices, such as STBs, then the server can send it a random key or key seed. The advantage of this approach is guaranteed uniqueness, but it carries a risk of key discovery in transit.

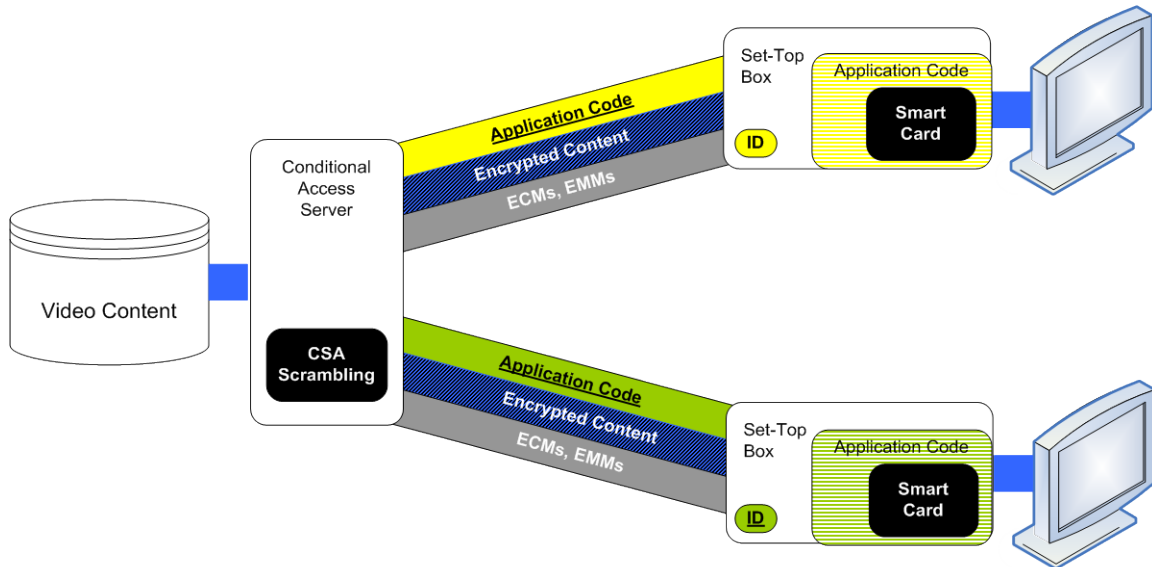


Figure 5: With individualization, code is bound to each STB through unique device IDs.

If the device is one whose hardware configuration can vary, like a PC, then a common technique is to compute a pseudo-unique ID based on hardware components. This technique dates back to the late 1980s. Microsoft uses the technique to generate so-called hardware hash codes in its Product Activation antipiracy technology for software products such as Windows and Office. Hardware hash codes are computed from several different hardware elements, and they behave according to an essential property of hash functions: that it is impossible to recover the original data from the hash value, thus preserving the privacy of the original data.

Once a unique ID has been established for the client device, the code can be individualized using a method that takes the ID as input. If the ID is not stored in secure hardware, it can be hidden using software techniques such as whitebox cryptography, described below.

We discuss more about this under code diversity below.

### Renewability

Content security systems must be renewable in order to stay secure over time. As discussed, older content security systems suffered from a lack of renewability because they worked on physical media (e.g. DVDs, Blu-ray discs) or on devices that could not be guaranteed to connect to a network.

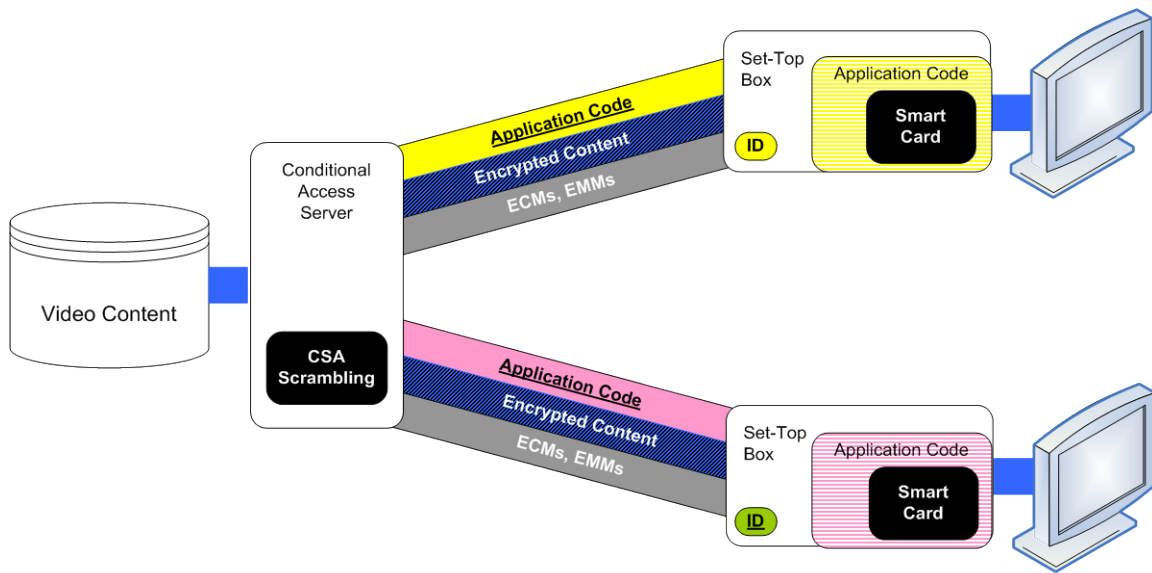


Figure 6: Renewability enables a client device to get different code over time. The most effective techniques involve renewing algorithms, not just keys.

Content protection systems for cable and satellite TV have had the luxury of assuming reliable network connectivity, so such schemes have pioneered renewability. Only recently have content protection schemes for PCs and portable devices been able to take advantage of presumed connectivity and provide renewability; Marlin and Microsoft PlayReady are two examples.

We saw on p. 11 above how lack of key renewability has hurt various content protection schemes, such as CSS for DVDs, HDCP for home device linking, and Windows Media DRM 7 for downloaded files. Content protection schemes for DVB systems have renewed keys through EMMs since the mid-1990s.

Yet key renewability itself has limited value. As long as a hacker has a method for discovering keys, he can continue to do so. There are two ways of foiling this: code instance renewability and algorithm renewability.

***As long as a hacker has a method for discovering keys, he can continue to do so. Code and algorithm renewability are ways of foiling this.***

With today's abundance of fast memory and processors, it's also possible to renew entire algorithm implementations. Widely-used crypto algorithms such as AES, ECC, and RSA can be fed parameters that implement code diversification (see below), so that an implementation on one client device is not necessarily the same as the implementation on another.

## Diversity

*Diversity* means installing different code in different devices so that a hack to one device does not necessarily work on another device of the same type, thereby diminishing both the impact of hacks and the incentive to develop them. Diversity overlaps with individualization to some extent.

### *Diversity diminishes the impact of hacks and the incentive to develop them.*

There are two types of diversity: *spatial* and *temporal*. Spatial diversity means that at any given moment in time, there will be different instances of functionally identical code on different devices of a given type.

A simple example of spatial diversity would be to have a set of functionally equivalent pieces of code and choose which one to deploy, based on a seed value such as a random number or device ID. Variations in code can include different combinations of control flow paths. The seed value would be used as input to a transformation routine that selects a particular instantiation of code before it is run.

Microsoft used a variation of this technique called the Black Box for crypto algorithms in one of its DRM systems back in 2001: the server created a new crypto algorithm for each instance of the PC client software by using parameters to generate symmetric algorithms for content encryption and asymmetric algorithms for protecting the symmetric keys. The result was a set of possible crypto algorithms to choose from, but it was unlikely that Microsoft provided a totally unique set of algorithms for each device. Microsoft has since incorporated this technique into its individualization scheme for Windows Media DRM as the Individualized Blackbox Component (IBX).

The most effective spatial diversity techniques have the capacity to create unique code for each device. This foils so-called scripting attacks, in which a hacker runs a code script automatically in such a way as to modify client code, in order to (for example) lengthen a user's subscription to a service. If the code is different on different client devices, then scripting attacks will not work.

Temporal diversity means installing code in a device that changes over time. This requires renewability. A general technique that facilitates both types of diversity is to use virtual machines. This is possible with today's client devices, with their larger amounts of memory and faster processors, and with high-capacity flash memory smart cards. With virtual machines, it's possible to download byte code into client devices more efficiently than producing and downloading entire application images into devices.

## Whitebox Cryptography

As we have seen, hiding crypto keys in software has been a species of what crypto experts call "security by obscurity," destined to fail if used by itself. Software key hiding techniques used in Windows Media DRM, AACs, Apple's FairPlay for iTunes, and other software-only content protection schemes have all been defeated.

Yet a new technique for software crypto implementations called *whitebox cryptography* has the potential to make standard crypto algorithms implemented in software sufficiently secure. The origin of whitebox cryptography was work that Stanley Chow and others did at Cloakware (now Irdeto) in the early 2000s.

Traditional implementations of cryptographic algorithms in software work on the assumption that hackers can only feed inputs (ciphertext) into the system and observe output (cleartext), or perhaps vice versa; these are known in the literature as known plaintext attacks (KPA) and known ciphertext attacks (KCA) respectively. These techniques depend on keeping the crypto implementation away from hackers' prying eyes, lest they are able to examine and reverse engineer it to discover encryption keys.

Developers have used methods such as *code obfuscation* to guard against such hacks. Code obfuscation refers to a wide range of techniques for "disguising" code so that it is difficult to reverse engineer. Some of these are automated techniques for transforming code, such as using device IDs to transform code or data; these are supersets of the individualization techniques describe above.

Whitebox cryptography eschews the idea of hiding or obscuring the functionality of code (i.e. putting it in a "black box") – specifically the code that implements crypto algorithms.

***Whitebox cryptography relies on large tables that make hacking through examining code harder than brute-force known-plaintext attacks.***

---

Instead, whitebox cryptography starts with the radical premise that a hacker can see and observe all of the code that implements the crypto algorithm, and relies on transformation techniques that make key discovery through code examination harder than brute-force attacks on the crypto algorithms. Whitebox techniques transform an algorithm implementation into a series of tables, along with code that implements the algorithm as a series of table lookups. These are far less efficient to traverse than it would be to reverse-engineer a standard code implementation for key discovery. Whitebox implementations also involve ways to inject randomness into the table-generation process to add both spatial and temporal diversity.

Whitebox implementations are available for most popular crypto algorithms. Chow et al at Cloakware created whitebox implementations of both AES<sup>11</sup> and DES<sup>12</sup> in 2002. Implementations of public-key algorithms such as RSA and ECC became available around 2007.

Figure 7 compares black and white box techniques. In the left-hand drawing, a hacker can use a brute force method of trying keys until he hits on the correct one, or he can try to defeat blackbox techniques such as code obfuscation, tamper-proofing, or individualization. The right-hand drawing shows whitebox cryptography, where it is less efficient to try to traverse the whitebox tables than to try a brute force attack.

The disadvantage of whitebox cryptography is that it requires much more memory than traditional techniques, even those that use code protection methods that substantially increase code size. For example, some protection methods install extra code that monitors execution for suspicious behaviors, such as the Guards in software protection tools from Arxan Technologies. This adds incrementally to the code size, whereas whitebox techniques increase code size dramatically. A whitebox implementation of a given algorithm will also run much slower than its blackbox counterpart. Therefore whitebox crypto was mainly of theoretical interest ten years ago.

---

<sup>11</sup> S. Chow et al, *White-Box Cryptography and an AES Implementation*. In Proceedings of the 9th Annual Workshop on Selected Areas in Cryptography (SAC'02), Aug. 15-16, 2002. Available at <http://www.cs.colorado.edu/~jrblack/class/csci7000/s05/project/oorschot-whitebox.pdf>.

<sup>12</sup> S. Chow et al, *A White-Box DES Implementation for DRM Applications*. In Proceedings of the ACM DRM 2002 Workshop, Oct. 15, 2002. Available at <http://crypto.stanford.edu/DRM2002/whitebox.pdf>.

But thanks to Moore's Law, whitebox crypto is now more practical to implement. It is beginning to appear in an increasing number of DRM implementations for PCs as well as advanced portable platforms such as devices based on Google's Android operating system, and it is known to satisfy the key-protection aspects of robustness rules for software implementations of many modern DRM systems. Thus whitebox crypto, with its automated code transformation, is particularly amenable to cross-platform implementation.

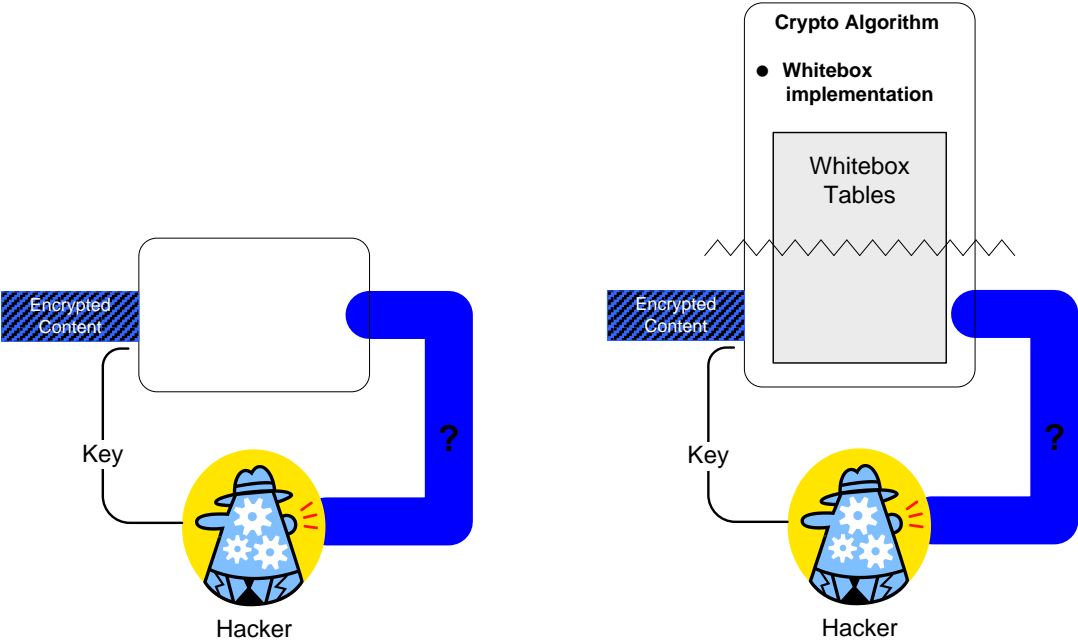


Figure 7: Blackbox and whitebox implementations of cryptographic algorithms.

### Security Lifecycle Services

Content security goes beyond code on servers and clients. It extends to elements such as the content provider's business model(s), content owners' strategic concerns in a world of proliferating delivery channels and platforms as well as unabated copyright infringement, and users' choices and expectations. Content security vendors should work with content service providers in important ways that complement the technological methods described above.

First, it is necessary to consider the entire threat model when designing a content security implementation. For example, operators and vendors should together consider the entire media path, including receivers of the content and other devices that could connect to it. Likely behaviors of normal users as well as hackers should be considered; the latter should be thwarted while the former are enabled. The result of this analysis is a threat model that can be used to plan an implementation that makes as much sense in the future as it does now.

After a system is deployed, the operator and vendor should monitor various potential sites of suspicious activity. In addition to monitoring the system itself and learning from data to improve proactive protections, content security vendors should be prepared to act as “intelligence agents,” monitoring websites, Twitter feeds, forums, and so on to be one step ahead of hackers.

Finally, techniques outside of the content protection technology can be used to keep the entire system secure on an ongoing basis, such as keeping system logs for audit purposes and maintaining physical security of the server infrastructure.

## Beyond the Server and Client: Emerging Techniques

Emerging techniques for content protection go beyond the media path from servers to receiving devices: they involve detecting unauthorized content sent over the Internet and/or stored in file-sharing networks.

### Content Identification

Various techniques for monitoring and analyzing Internet traffic to detect suspicious activity such as unauthorized file-sharing have existed, but their effectiveness has not been that great: among other shortcomings, such techniques have been easy to evade.

Therefore techniques have been developed that focus on identifying content “in the wild” and tracing it back to the user or device that put it there. These are special cases of techniques known collectively as *content identification*.

*The key to finding unauthorized content “in the wild” is to identify it. The two basic techniques for identifying content are watermarking and fingerprinting.*

---

There are two primary ways to identify content: *watermarking* and *fingerprinting*. Watermarking (or digital watermarking) is the earlier of the techniques, generally dating back to the mid-late 1990s. Just as watermarks in paper (money, postage stamps, stationery, etc.) involve images embedded in the paper, digital watermarks involve data embedded into digital content. Although some watermarks are meant to be visible, for these purposes we assume they are embedded into the “noise” portions of digital content so that humans can’t perceive any difference when the content is rendered (displayed, played, etc.). Watermarking began as a technique for digital images and has expanded to audio and video.

Watermarking, shown in Figure 8, can be used to identify content simply by embedding the identity of the content or its owner as a *payload*, which is the data embedded into the content as a watermark. Watermark designs trade off among several characteristics, including:

- **Capacity:** the amount of data that the payload can hold. 100 bytes is often used as a rule-of-thumb measure of capacity. This is not much data, so the payload is often used to store an index to a database rather than data itself.
- **Imperceptibility:** the lack of effect on the user’s perception of the content.
- **Robustness:** the watermark’s imperviousness to distortion or obliteration if the content is converted to another format, converted to analog, or transformed in some way (cropped, edited, etc.).
- **Security:** the watermark’s imperviousness to removal without visibly or audibly damaging the content.

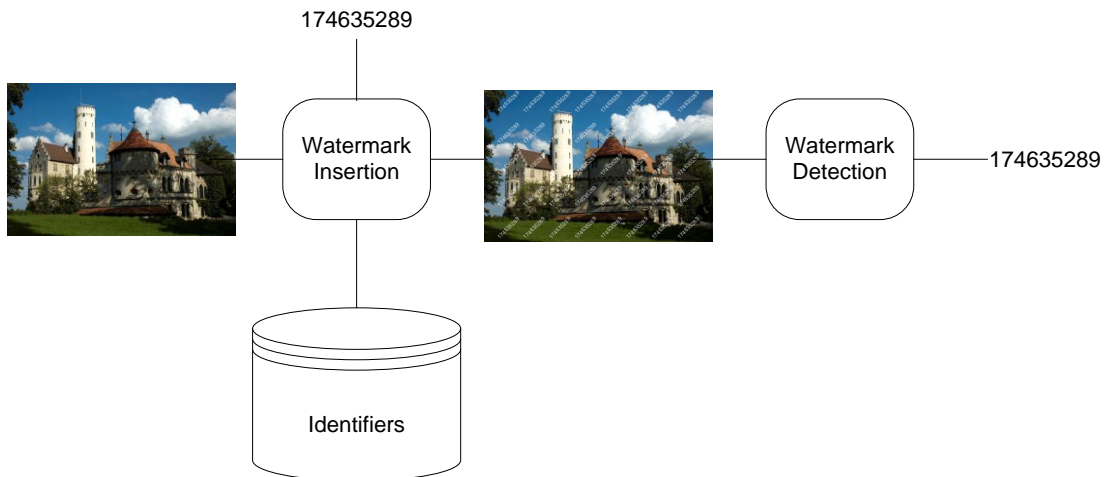


Figure 8: Watermarking entails embedding an identifier as the payload, then extracting the payload later.

Fingerprinting, shown in Figure 9, is named as an analogy to taking fingerprints, such as at a crime scene, to determine identities of people who were at the scene. It is a means of identifying content by “taking its fingerprint” and looking the fingerprint up in a database of known fingerprints.

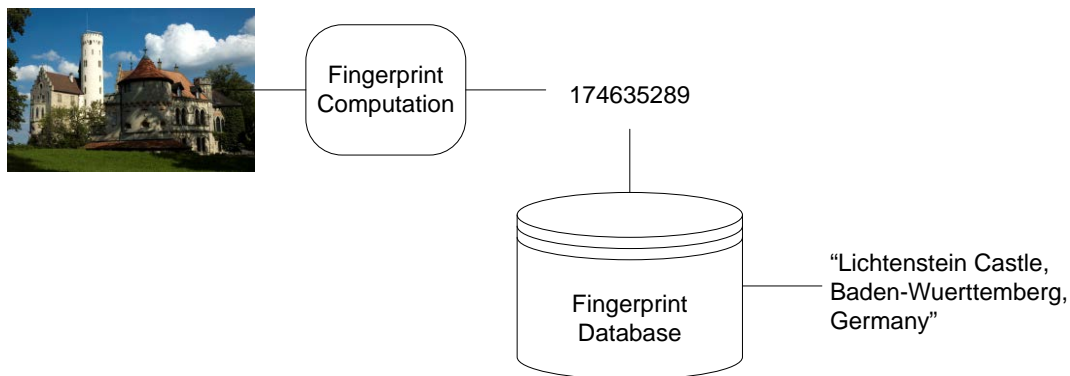


Figure 9: Fingerprinting uses an algorithm to compute characteristics of the content and output one or more numbers, which are looked up in a database of fingerprints.

A fingerprint is a set of numbers that encapsulates essential characteristics of the content, such as images and audio, and forms an identifier for the content. In mathematical terms, fingerprint algorithms are special cases of hash functions – specifically those with the property that perceptibly similar data sets (e.g. two different digital recordings of the same TV show or music track) compute the same hash value instead of different ones. In other words, whereas standard hash functions are designed to detect data that has been

*altered*, even slightly, fingerprint algorithms are designed to detect data that is *perceptibly similar* to known data<sup>13</sup>.

Fingerprinting technology is slightly more recent than watermarking; the first important developers of the technology were Audible Magic and CDDDB (now Gracenote, owned by Sony Corp.) in 1999-2000. The first several years of fingerprinting technology were focused on audio (music); then starting around 2006, video fingerprinting solutions began to appear. Google developed its own fingerprinting system for YouTube in 2007. Fingerprinting techniques have also been used for digital still images.

Fingerprinting and watermarking are complementary. The biggest differences are these:

- Watermarks must be embedded into the content in order to be used, whereas fingerprinting requires no such preparation.
- Fingerprinting is really an “educated guess” of a content file’s identity. It is not 100% accurate, as watermarking can be (if the payload is a content identifier), yet it is accurate enough that major content owners have generally accepted its use.
- Watermark payloads can contain any data at all (up to capacity limitations). Fingerprints can only identify the content.

Watermarks that contain content identifiers (e.g. ISAN numbers for video content) are not particularly helpful in preventing content misuse. Similarly, fingerprinting can only identify the content, not where it has been or what user has downloaded or uploaded it. However, techniques based on these technologies are emerging that can be helpful in catching individual infringers.

#### Session-Based Watermarking

It is possible to embed data identifying the device that receives or downloads content as a watermark payload. The term most often used for this is *session-based watermarking*, also known as *transactional watermarking*. This way, files “in the wild” can be run through watermark detectors, and the resulting payload can indicate a device or user ID. However, such data might be too large to fit in a watermark payload, so instead, a serial number can be used that causes a database lookup; this approach has been called *media serialization*.

#### ***Session-based watermarking makes it possible to trace a file found on the Internet to the device that downloaded it.***

---

Several vendors have offered different transactional watermarking techniques over the past few years, including Civolution and Verimatrix. Hollywood studios currently require session-based watermarks in addition to encryption for digital distribution of early release window content.

However, in other areas, session-based watermarking has been slow to take off, mainly for two reasons: efficiency and privacy. In session-based watermarking, a different watermark must be generated for every stream or download. The watermarks can be generated either on servers or clients.

---

<sup>13</sup> For this reason, some researchers refer to fingerprinting as *perceptual hashing*.

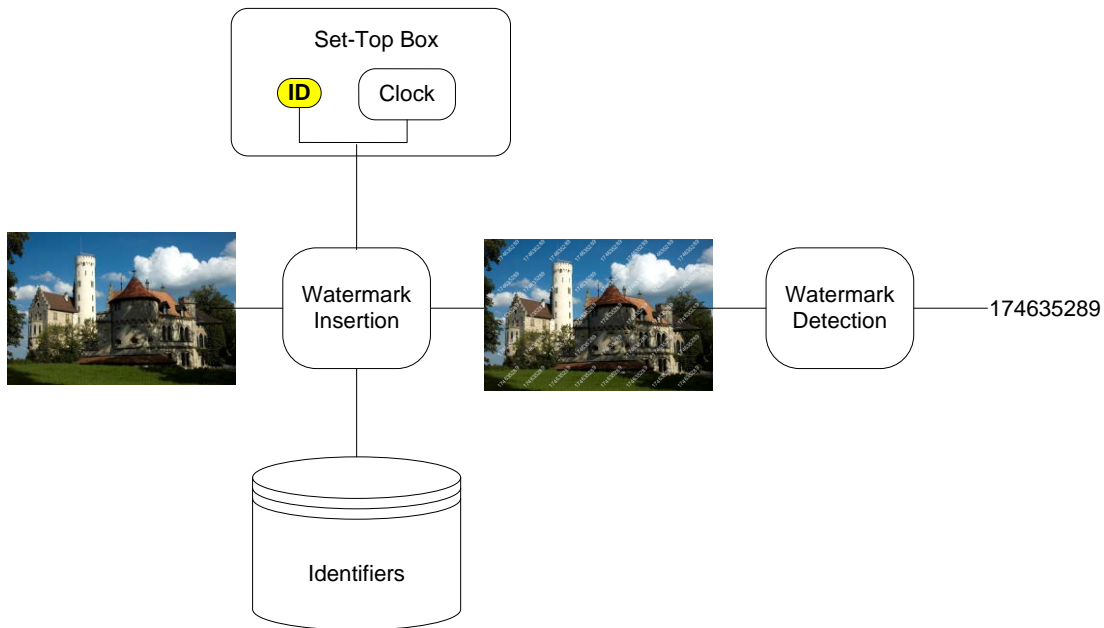


Figure 10: Session-based watermarking scheme running on a STB computes a watermark based on a device ID and a timestamp.

Inserting session-based watermarks at the head end requires much more server capacity than watermarking schemes that insert the same watermark in every download, such as MP3 music files which often contain watermarks identifying the retailer (e.g., Amazon.com or Wal-Mart).

Otherwise, session-based watermarks can be inserted on client devices. In that case, the functionality for establishing session IDs and inserting them as watermark payloads must be tamper-proofed, and additional computational power or memory may be needed to do the insertion efficiently enough.

Both of these cases present problems for pay TV operators and other service providers, because they add to costs. Yet Irdeto recently announced a technique called TraceMark that does server-side session-based watermark insertion in a more efficient way: TraceMark computes two watermarks and interleaves them differently for each video stream. This technique results in a large number of different watermarks at a fraction of the computational cost required to generate different watermarks from scratch. TraceMark can be used with a variety of watermarking algorithms.

The other concern about session-based watermarking is privacy, which applies more to downloads than to streams or broadcast content. Some service providers and content owners are concerned that users will object if personally identifiable information (PII) is inserted in their content. A related concern is misdirected infringement accusations: if one user downloads a file, gives it to another user, and then the second user puts the file up on a file-sharing or BitTorrent site, then the watermark traces it to the first user, who has not done anything wrong.

## Fingerprinting and Deep Packet Inspection

Content identified via fingerprinting can be linked to users who send it over networks by means of *deep packet inspection* – that is, implementing fingerprinting at the network infrastructure level. A fingerprint computation appliance that operates at this level can feed data to network monitoring tools that link the files being sent to the identity (such as IP address) of the sender.

*Deep packet inspection makes it possible to link content identified via fingerprinting with the sender, though with cost and efficiency burdens on network operators as well as privacy concerns.*

---

The big disadvantage of this technique is, of course, the cost and efficiency burden that it places on network operators. Nevertheless, network operators have been experimenting with it. For example, AT&T (in its role as an ISP) has experimented with technology from the fingerprinting vendor Vobile. There are also, as with session-based watermarking, privacy concerns about deep packet inspection.

Nevertheless, there are two reasons why network operators may be interested in implementing fingerprinting via deep packet inspection. First, some countries are beginning to enact laws that require ISPs to take responsibility for copyright infringement, such as France, UK, South Korea, Taiwan, and New Zealand – although so far, such laws are generally set up so that copyright owners must bring evidence of alleged infringement, i.e., ISPs are not held responsible for policing their own networks.

Second, ISPs are interested in differentiating their offerings in an increasingly competitive environment. This includes the ability to bundle premium content with network connectivity. Some major content owners are requiring network service providers to improve content security as a condition of licensing their content.

## Conclusion

In this white paper, we have explored the history of security technologies introduced into the pay television world since the adoption of the DVB standard in 1994. We have seen various security weaknesses that have led to hacks over the years in both digital conditional access and DRM technologies for content downloaded over the Internet.

We have also described various techniques that are considered state-of-the-art for protecting digital TV signals. These include generic technology developments such as elliptic curve cryptography (ECC), flash memory, and secure silicon. They also include techniques for client device (e.g., set-top box) security, such as individualization, renewability, spatial and temporal diversity, code obfuscation, and whitebox cryptography.

Yet these techniques are only effective if integrated into an end-to-end security scheme that encompasses the entire media path and includes monitoring, audit trails, and detection of unauthorized content “in the wild.”

The techniques described here are being successfully deployed by companies such as Irdeto for pay-TV operators worldwide. Because pay television is often at the forefront of content security techniques, participants in value chains for other types of digital content (music, e-books, etc.) and other delivery modalities (e.g. Internet downloads, physical media) may be able to learn from these techniques. The pay TV industry will make them robust and cost-effective so that related industries can benefit from them.

Finally, although vendors of pay TV security solutions expect that the latest set of techniques will minimize the impact of hacks and put an end to the cycle of hack-patch-hack-patch, the only true test of any security technology is the test of time. Security systems need not remain unbroken forever; they just need to be robust until the content delivery technology they support is sufficiently obsolete. Given the pace of change in digital content technology, that’s far short of forever.

### About Irdeto



Irdeto is the most innovative software security and media technology company in the world. Through its dynamic security and monetization technologies, the company allows new forms of distribution for broadcast/broadband/mobile entertainment, and for the world’s most popular app, eStores and consumer devices. Co-headquartered in Amsterdam and Beijing, Irdeto employs 1000 people in 25 locations around the world. It is a subsidiary of broad-based media group Naspers (JSE: NPN). Please visit Irdeto at [www.irdeto.com](http://www.irdeto.com).

## About the Author

**Bill Rosenblatt** founded GiantSteps Media Technology Strategies in 2000. He is the author of several books, including *Digital Rights Management: Business and Technology* (John Wiley & Sons, 2001), the chapter “Digital Rights and Digital Television” in *Television Goes Digital* (Springer, 2010), and several white papers on digital rights and content management technologies. He has served as a technical expert in litigation and public policy initiatives related to digital copyright. He is editor of the blog Copyright and Technology (<http://copyrightandtechnology.com>) and Program Chair of the Copyright and Technology Conference.

## About GiantSteps Media Technology Strategies



**GiantSteps Media Technology Strategies** is a management consultancy focused on the content industries that help its clients achieve growth through market intelligence and expertise in business strategy and technology architecture. GiantSteps' clients have included branded content providers, digital media technology vendors ranging from early-stage startups to Global 500 firms, and technology public policy entities in the United States and Europe. For more information, please visit [www.giantstepsmts.com](http://www.giantstepsmts.com).