Turning Teenagers into Stores

Srijith K. Nair, Bruno Crispo, and Andrew S. Tanenbaum, Vrije Universiteit
Ron Gerrits, Inovia

Paradiso is a prototype of a system that lets consumers contact content providers to buy songs and videos—and to buy optional content-resale rights. In essence, the scheme would turn customers into content distributors, provide wider reach, and free up content providers' bandwidth. However, such an architecture requires strict security precautions and interoperable digital rights management standards among player manufacturers and content providers.

The intersection of computers, the Internet, music, and teenagers has uprooted the music industry. For more than a century after Thomas Edison's 1877 invention of the phonograph, the industry sold singles or albums recorded on wax or plastic media to consumers in record and department stores. New media were introduced, including the 78-rpm wax single, the 45-rpm single, the 33-rpm LP record, and finally the Philips CD, but the business model stayed the same. With the invention and 1991 standardization of the MP3 psychoacoustic compression algorithm by engineers working on the European Union's Eureka project 147, the era of downloadable digital music was launched.

After the Fraunhofer Institute released the first MP3 encoder in 1994, many young music fans began to encode their audio CDs in MP3 format and store them on their computers' hard disks. Before the invention of MP3, storing music on a PC's hard disk wasn't practical because a single CD could take up to 650 Mbytes, and hard disks were smaller than 1 Gbyte at this time. But with tenfold compression possible with little quality loss, storing and playing music on computers skyrocketed.

It didn't take long before friends began exchanging music files over the Internet. Napster debuted in 1999, offering a central catalog of who had which songs, so people could directly copy songs from the remote hard disks of people they didn't know. Napster users thought of it as a wonderful new invention: peer-to-peer file sharing.

Unfortunately for them, people in the music industry didn't see it that way. They saw it as theft of their intellectual property, and they responded by suing Napster and closing it down. Decentralized services such as Kazaa and Grokster soon replaced Napster, and they were sued with mixed results. Then the music industry began suing individual teenagers for copyright violation, seeking maximum publicity when they settled out of court for thousands of dollars.

Eventually, it dawned on them that suing their own customers (especially children) wasn't a good business model. This led to the development of online music stores that let customers legally buy and download songs from the store's central server. The first major online-music seller was Apple with its hugely successful iTunes store (www.apple.com/itunes) and iPod player.

iTunes uses a completely centralized digital rights management (DRM) system called FairPlay, with users contacting an Apple server to buy and download music and authorize their usage. When Microsoft released its Zune player (www.zune.net) and online store in November 2006, it added a new feature lacking in iTunes: a limited ability for a user to transmit a song to a friend's Zune player offline, without having to contact the central Zune server. However, a user can only transmit a

 Published by the IEEE Computer Society 0018-9162/08/$25.00 © 2008 IEEE

song three times and store it for three days. If the friend likes the song, he must contact Zune's server to buy it. Figure 1 shows the two models.

By now, the music companies have come to realize that digital music is their friend (just as the movie studios eventually stopped suing VCR manufacturers and began releasing movies for rent). They also realize that many teenagers become aware of songs when they plug into a friend's music player and listen to music that way, a practice now known as "jack sharing." This knowledge has led some music executives to dream of turning teenagers into stores, legally reselling songs they've bought, a concept more prosaically called *superdistribution*.[1]

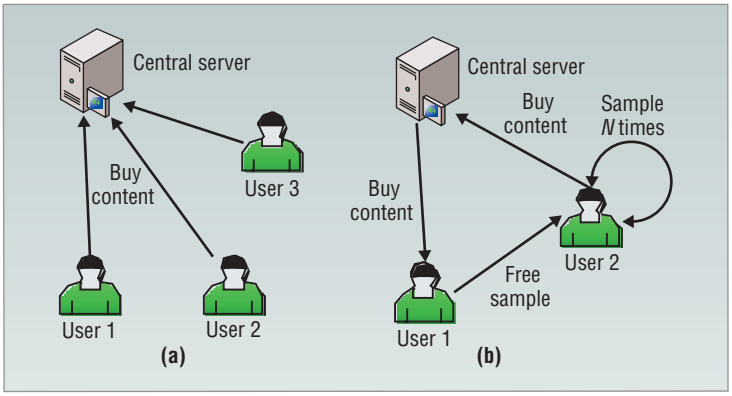

Figure 1. Music distribution models. (a) Apple uses a central store model to sell its iTunes. (b) Microsoft's Zune allows limited sharing of content.

A MODEL FOR RESELLING MUSIC

What we need is a scheme that turns willing customers into full-fledged resellers. Amsterdam's Vrije Universiteit has developed a system that could serve as a prototype.

Consider this scenario: Bob visits an online content provider like iTunes and buys a song for 99 cents. Having an inkling that the song will also be a hit with his friends, he buys the right to resell the song to nine friends for a total of $8.91, getting a 10 percent discount for buying 10 units. He pays the $8.91 in advance by credit card. Bob then hooks up with his friend Mark and tells him about the cool song he just got. After hearing the song using Bob's player, Mark decides to buy a copy. Bob sells Mark the copy (using the wireless link) for 95 cents, making a 6-cent profit.

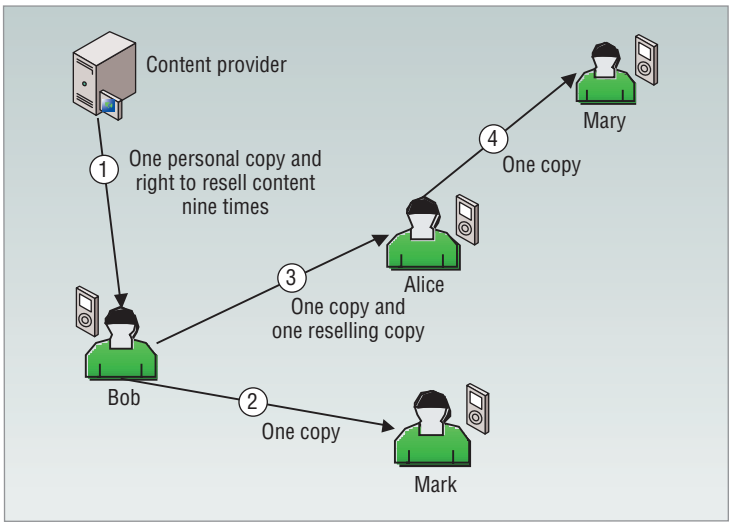

Figure 2. Content buying and reselling process flow.

On his way home, Bob meets up with Alice and sells a copy of the song to her. Alice tells Bob that her friend Mary might also be interested in the song, so she buys it and the right to resell it once, paying Bob $1.90. When Alice runs into Mary, she sells Mary the song for 97 cents. Figure 2 illustrates these transactions.

From the point of view of consumers like Bob and Alice, the benefit is evident. By acting as a reseller on behalf of the content owner, the consumer earns a profit per song sold. Mark and Mary also benefit by getting the song immediately and for less than the retail price. The content owner can reach a wider range of potential consumers, and, more importantly, the network formed between the consumers is more taste-targeted than any marketing campaign. Furthermore, by transacting many of the sales without involving the central distribution server, the content vendor's server and bandwidth requirements are greatly reduced, cutting associated costs.

Of course, the trick is to have the technology to enable the above scenario in such a way that content doesn't become freely available.

THE TECHNOLOGY

Achieving interoperability between different manufacturers' players and providers' content requires IEEE or other standards. Player manufacturers would need to design according to these specifications and undergo compliance testing and accreditation from a certification authority. Upon passing the accreditation tests, the CA would certify the manufacturer by signing its public key. The manufacturer would in turn sign the unique public key of each player it produces, thus producing a chain of trust to identify all specification-compliant players. Each player would contain its own certified public and private keys, the manufacturer's certified public key, and the CA's public key.

The manufacturer needs to store the player's private key in secure hardware to prevent direct unmediated access. All private-key operations must be performed within the secure hardware in a controlled manner. Among other functions, the secure hardware must perform asymmetric and symmetric key encryption and decryption and collision-resistant hashing.

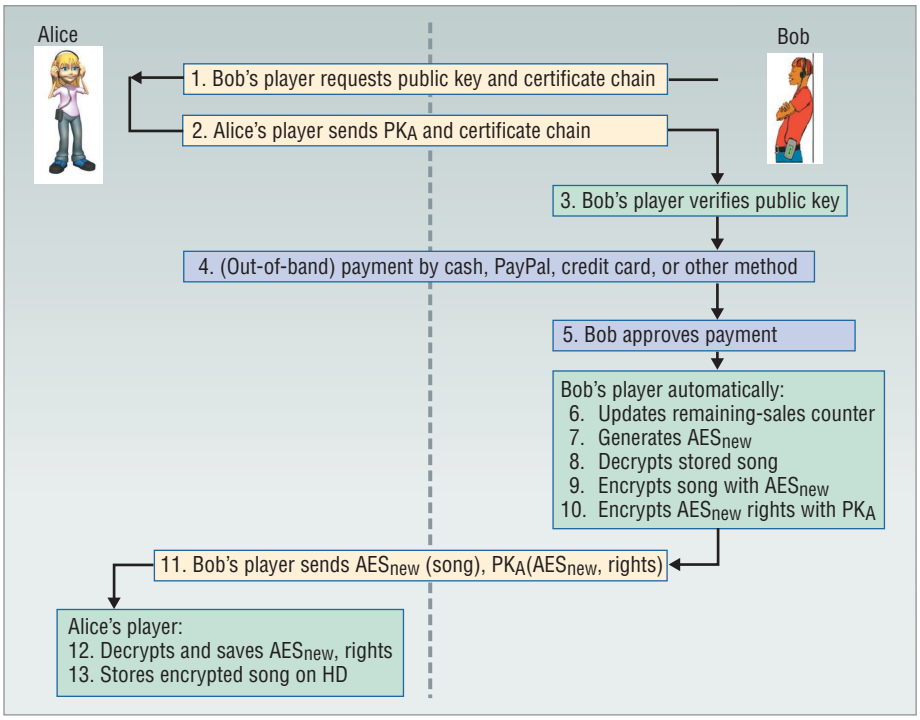

Figure 3. How Bob sells Alice a song. The yellow boxes are messages, the green ones are steps the cryptochips perform, and the blue ones are manual steps.

Since an attacker or misbehaving consumer might misuse content, it's stored encrypted. In addition, a cryptochip (preferably soldered to the player's motherboard or to a PCI plug-in card) performs sensitive operations. The cryptochip must contain a CPU, nonvolatile memory for key storage, and some working RAM. In this way, the player can issue a command to the cryptochip saying, "Fetch the encrypted song at memory address 0x122400, decrypt it with symmetric key #4 in your internal list, and generate audio on your output pins."

In this way, the plaintext music is never released outside the cryptochip. A chip like the trusted platform module (www.trustedcomputinggroup.org/groups/tpm) already provides some of these features. We feel that technology is advanced enough to expect implementation of such features, given enough financial incentives. With the content never appearing in plaintext outside the cryptochip, the security requirements on the rest of the software become much less stringent.

SYSTEM ARCHITECTURE

In our architecture,[2,3] a consumer contacts a provider to buy a song, video, or other content and optionally the right to resell it *N* times. The request message contains the player's public key. Once the consumer has paid for the content and the rights, the provider encrypts the content on the fly with a newly generated symmetric Advanced Encryption Standard (AES) key and sends the encrypted content to the consumer's (now called the reseller's) player.

The content provider also sends the symmetric key encrypted with the player's public key and the rights the reseller bought, expressed in a suitable language. It signs the rights with a secret key to prove they're valid. The secure hardware on the consumer's player checks the integrity of the content and the rights. If valid, they're stored on an insecure hard disk or flash memory. Since the rights information is signed by the content provider and its hash is stored in a secure memory, the owner can't tamper with it.

Process steps

Each time a reseller such as Bob wishes to resell the content, the cryptochip first checks to see if the maximum number of sales the license defines has been reached. If not, it goes through the following steps, which Figure 3 illustrates. In Step 1, Bob asks a buyer such as Alice to send her public key, PK_A, and a certificate chain rooted at the CA over the wireless link to Bob's player. In Step 2, Alice's player sends the PK_A and certificate chain. In Step 3, Bob's cryptochip verifies that Alice's claimed public key is in her certificate, that the player's manufacturer signed the certificate, and that the CA approved the manufacturer.

If all goes well, Bob's player now knows that Alice's player has been certified as compliant. After all, although Alice can easily generate a private-public-key pair, she can't produce a certificate chain back to the CA guaranteeing that the key is authentic, and without this authenticity, Bob's cryptochip won't allow the transaction to proceed.

In Step 4, Alice pays Bob using cash, PayPal, credit card, or another agreed-upon means. This step is out of band and not part of the protocol. In Step 5, when Bob is satisfied with the payment, he pushes a button on his player to approve the sale. In Step 6, the cryptochip first updates the number of sales remaining and keeps this counter in its secure internal memory. Then, in Step 7, it generates a fresh symmetric key, AES_{new}.

Using the existing stored per-song symmetric key, in Step 8, the cryptochip reads the song from main memory and decrypts it, and in Step 9, reencrypts it with AES_{new}, and puts the newly encrypted song elsewhere in insecure RAM, leaving the original intact.

In Step 10, Bob's cryptochip encrypts the new symmetric key, AES_{new}, with Alice's valid public key and puts it

in RAM as well. Bob can steal the song and the key from RAM, but it won't do him any good as they're encrypted, the song with a symmetric key he doesn't know and the song key with Alice's public key. Furthermore, he can't get at the "remaining-sales" counter, which is kept safely in the cryptochip's internal nonvolatile memory.

Next, in Step 11, Bob's player sends Alice the AES-encrypted song and encrypted AES key for this song. In Step 12, upon receipt of the message, Alice's cryptochip decrypts it and saves the song key, AES_{new}, internal to itself; in Step 13, it saves the encrypted song on the (insecure) hard disk or flash memory.

Critical events

Steps 6-10 are performed as a single atomic transaction, but if Bob's player is switched off between steps 6 and 11, maliciously or otherwise, he loses one resale right and must deal with an unhappy customer who didn't get the song she paid for. Other critical events are an accidental communication breakdown between the reseller and the consumer while the transaction is in progress or when the reseller cheats the consumer by delivering a bogus song. To resolve all these situations, a "recovery subprotocol"[3] lets the consumer contact the content provider directly to resolve the issue.

It's important to note that the resale is offline. Neither Bob nor Alice has to contact the content provider since Bob has already paid for Alice's copy of the music (as well as the eight unsold copies) in advance. By using teenagers as salespeople, the content provider saves on computing power and bandwidth costs.

What happens if Bob can't find eight more friends who want the song? The publishers of books, magazines, and newspapers have precisely the same problem, and they generally allow their sales outlets to return unsold stock for credit to encourage them to have an ample supply on hand. Of course, the publisher can rescind any quantity discount granted initially when the vendor returns the unsold copies. Following this tradition, music vendors are likely to follow suit, but that's their business decision to make.

When Alice wants to listen to her newly purchased song, the cryptochip in her player extracts its symmetric key, AES_{new}, stored in its internal memory and fetches, decrypts, and plays the song one block at a time. In this way, the bulk data—the songs—are stored in the large cheap memory, with each song encrypted with a unique symmetric (AES) key.

SECURITY CONSIDERATIONS

However secure and foolproof we assume a system to be, experience has shown that all it takes is a single weak link to compromise its security. Our scheme assumes that certified players behave in the stipulated manner and that they follow the protocols correctly. However, it might be possible to crack a player using out-of-band methods, such as using an electron microscope to read the keys in the cryptochip's internal nonvolatile memory.

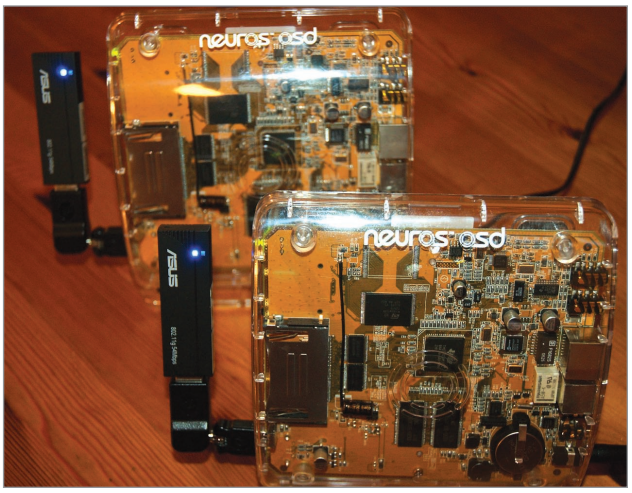

Figure 4. Prototype implementation using Neuros OSD boards.

Watermarking and traitor-tracing techniques[4] can be used to identify such compromised players. Once identified, the compromised player's identity (public key) is added into a player revocation list. The system can push this list to each consumer's player the next time it connects to a content provider. Other researchers have proposed various ways to minimize the size of such lists.[5] As an enhancement, the players could also exchange revocation lists when they exchange content. A compliant player is designed to refuse communication with any player listed in the revocation list.

Of course, our system also suffers from the "analog hole" problem. An attacker can always record the content with a microphone while it's being played and redistribute it in an uncontrolled manner. There's no definitive solution for this problem; however, the degradation in the quality of the copy obtained through analog recording could be an attack deterrent.

It's important to note that our system doesn't introduce any new vulnerability. These attacks also apply to current players that don't allow controlled peer-to-peer distribution.

PROTOTYPE IMPLEMENTATION

We've implemented Paradiso (www.few.vu.nl/~srijith/paradiso), a system prototype, using a $230 Neuros development board (http://wiki.neurostechnology.com/index.php/OSD_Beta), representative of what's found in mobile music players. Shown in Figure 4, this board has a TI 200-MHz ARM926, 120-MHz C54x DSP processor specifically developed for multimedia applications, 64 Mbytes of SDRAM and 10/100 Mbps Ethernet port, among others. The board runs a modified version of the Linux 2.6 kernel. We used OpenSSL libraries for cryptographic support and software techniques for atomic actions.[6]

The developer boards acted as compliant players. However, since we couldn't obtain a developer board with a suitable cryptochip and secure store, we used a software layer to emulate the hardware security layer. We believe that once the interface and protocols have been defined, implementing them on another (secure) processor wouldn't be difficult.

Experiments performed with our prototype show that it takes around 10 seconds to perform steps 6-13 for a 5-Mbyte file. Performance measurements also show that the music file's quality doesn't suffer from the lag due to the decryption steps. While the prototype implements the cryptographic steps in the software, a production unit will implement them in the hardware, thus we can expect a speedup and better performance.

Microsoft's Zune took the first baby step toward implementing our proposed system by letting users forward songs to friends. However, the similarity ends there. The recipient still must contact the content provider to purchase the content and associated license. Zune's existence is an indication of the digital medium's potential, as well as content owners' and player manufacturers' receptiveness to explore new avenues to widen their reach.

Although we designed our prototype to generate revenue, DRM technology can easily be extended to serve the needs of consumer-produced digital content. For example, a band could produce and release a song under one of the "noncommercial" Creative Commons licenses and upload it to a content provider as a way to promote its new album. The trusted player, on noticing the song's license, would let the song be exchanged for free.

Similarly, Bob could use the same technology to share the latest video clip he's shot. One of Zune's perceived shortcomings is that irrespective of the origin and license of the content a user exchanges with another, the content is deleted after three plays or days. It's evident that designers incorporated such limitations to prevent using Zune as a new illegal peer-to-peer medium. However, in this age of consumer-generated content, a DRM scheme shouldn't deny copyright owners the right to give away content for free if they so choose. Just imagine the fuss if all computers automatically deleted all free software after three days.

The goals of a Paradiso-like system, however, aren't realizable without some mind-set change. As of now, every player manufacturer uses DRM technology that's not interoperable with other manufacturers. A Paradiso-like architecture would require major manufacturers and content owners to use interoperable DRM technology standards. The success of industry-wide specifications like the mobile industry's Open Mobile Alliance indicates that such an alliance is possible, given strong enough incentives.

Acknowledgments

The work described in this article has been supported by NWO project Account 612.060.319. We thank Bogdan C. Popsecu, Chandana Gamage, Mohammad T. Dashti, and Hugo Jonker for their assistance.

References

1. R. Mori and M. Kawahara, "Superdistribution: The Concept and the Architecture," *IEICE Trans.*, E73, no. 7, 1990, pp. 1133-1146.
2. S.K. Nair et al., "Enabling DRM-Preserving Digital Content Redistribution," *Proc. 7th IEEE Int'l Conf. E-Commerce Technology*, IEEE Press, 2005, pp. 150-158.
3. M.T. Dashti, S.K. Nair, and H.L. Jonker, "Nuovo DRM Paradiso: Towards a Verified Fair DRM Scheme," *Proc. Int'l Symp. Fundamentals of Software Eng.* (FSE 07), Springer-Verlag, 2007, pp. 33-48.
4. B. Chor, A. Fiat, and M. Naor, "Tracing Traitors," *Proc. Advances in Cryptology*, (CRYPTO 94), LNCS 839, Springer-Verlag, 1994, pp. 257-270.
5. D.A. Cooper, "A More Efficient Use of Delta-CRLs," *Proc. 2000 IEEE Symp. Security and Privacy*, IEEE Press, 2000, pp. 190-202.
6. R. Gerrits, "Implementing a DRM-Preserving Digital Content Redistribution System," master's thesis, Vrije Universiteit, 2006.

Srijith K. Nair *is a PhD student in the Department of Computer Science at Vrije Universiteit, Amsterdam. His research interests are information flow control, policy enforcement, system security, and privacy. He received an MSc in computer science from the National University of Singapore. Nair is a member of the IEEE and the ACM. Contact him at srijith@few.vu.nl.*

Ron Gerrits *is the owner of the Web-development firm Inovia. He obtained an MSc in Internet and Web technology from Vrije Universiteit. Contact him at msc.r.gerrits@inovia.nl.*

Bruno Crispo *is an associate professor in the Department of Computer Science at Vrije Universiteit and at the University of Trento, Italy. His research interests are networks, distributed systems, cryptography, and security protocols. Crispo received a PhD in computer science from the University of Cambridge, United Kingdom. He is a member of the IEEE. Contact him at crispo@cs.vu.nl.*

Andrew S. Tanenbaum *is a professor in the Department of Computer Science at Vrije Universiteit. His research interests are reliable operating systems and security. He received a PhD from the University of California, Berkeley. Tanenbaum is a Fellow of the IEEE and the ACM and a member of the Royal Netherlands Academy of Arts and Sciences. Contact him at ast@cs.vu.nl.*