FluidMedia: an offline peer-to-peer media transaction system

Kan Zhang, Cyril Brignone, Christophe Gouguenheim,
Fred Kitson, Salil Pradhan, John Recker, Bill Sera
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2002-342
December 11th , 2002*

The FluidMedia system provides a convenient means to legally transfer copyrighted digital media. Its ability to support off-line sales transactions and authentication of media access rights allows system users great flexibility in the way they access and distribute media, yet still provides strong control over piracy for content distributors. Furthermore, the FluidMedia system supports transaction policies that provide economic incentives to system users to distribute media.

* Internal Accession Date Only
Approved for External Publication

FluidMedia: an offline peer-to-peer media transaction system

Kan Zhang, Cyril Brignone, Christophe Gouguenheim,
Fred Kitson, Salil Pradhan, John Recker and Bill Serra

Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA

Abstract

The FluidMedia system provides a convenient means to legally transfer copyrighted digital media. Its ability to support off-line sales transactions and authentication of media access rights allows system users great flexibility in the way they access and distribute media, yet still provides strong control over piracy for content distributors. Furthermore, the FluidMedia system supports transaction policies that provide economic incentives to system users to distribute media.

1 Introduction

In 2002, Michael Green, the president of the Recording Industry Association of America (RIAA) gave an impassioned speech to the national TV audience watching the Grammies music award show declaring that illegally downloaded music from the Internet was a significant threat to the music recording industry.

This followed prolonged legal action by the RIAA against Napster, the first large scale system to support free downloading of copyrighted digital media, that resulted in effectively closing down Napster's business. Although more recently some artists have publicly stated that the RIAA is overstating the extent of the problem[1], there can be no doubt that significant numbers of people are illegally downloading significant amount of digitised music. Furthermore, the increased availability of broadband access in the home and tools such as DVD burners suggests that this trend will proliferate into other forms of copyrighted digital media.

It is our belief that while the zero cost of the media contributes to the popularity of peer-to-peer digital media exchange systems such as KaZaA[1], Morpheus[2] and others, the convenience offered by these systems is also a significant impetuous for their use. Yet there is no significant competing legal means of quickly accessing music content. Sure, one can travel to a CD retailer who may or may not stock a CD of interest, or one can ask an on-line retailer to ship the CD overnight for some significant shipping surcharge, but these delivery mechanisms clearly do not compete with the convenience offered by the peer-to-peer systems which can often offer access to a track or CD of interest in minutes. This is particularly significant for music, which is often an impulse purchase.

FluidMedia is a peer-to-peer system for quickly and legally transferring digital media; those transfers can be made offline. Nevertheless during each transfer a fee is paid to the provider of the media. Not only does the system provide a convenient means of immediately transferring music from one consumer to another, it also offers economic incentive for consumers to sell music and other digital media from one to another. The FluidMedia system allows the user to play entitled media stored on any FluidMedia devices, while maintaining a reasonable level of security to prevent widespread pirating of digital media.

2 The FluidMedia transaction model

FluidMedia supports secure payment and exchange of encrypted media files. Unlike online electronic payment schemes, however, transactions depend only on digital keys and tokens (FluidMedia system private virtual money) stored locally on *Smart Cards*[2], portable programmable devices designed for secure storage of data. As a result, transactions can be performed both between devices disconnected from the Internet infrastructure and with the more typical Internet based services.

FluidMedia separates the payment for media from the download of the media file. Media files are stored encrypted on devices. Each piece of media is associated with a unique media key. The media keys are stored on smart cards representing the digital rights the owners of smart cards have. Media keys are used as master keys to derive device-specific encryption keys for the encryption/decryption of media files on devices. The same media is encrypted differently on different devices. This ensures that cracking a device does not challenge the integrity of the whole system. Yet, the unique media key can be used to decrypt the associated media on any device. Buying a piece of media is

[1] www.kazaa.com

[2] www.morpheus.com

essentially buying the associated media key, while the encrypted media files can be freely distributed or even preloaded to any FluidMedia devices. A user can have any number of encrypted versions of a song stored on a variety of devices, for the smart card ensures that only one version of the song can be played at any one time on any device.

Every FluidMedia device can therefore act as a point of sale. One result is that non-traditional retail establishments can enter the media sales business. For example, a coffee shop can offer for immediate purchase a song playing over the shops music system. In addition to creating a market for a new type of consumer electronic device, this offers the media producers the opportunity to dramatically increase the outlets for their product. However, this increase in retail outlets is only the tip of the iceberg. The real increase in media sales will likely occur as a result of person-to-person transactions, much like the direct sales approaches pioneered by companies such as Avon Products.

FluidMedia provides incentives for a user to sell media to others as well. First, the system provides a seamless and easy mechanism for browsing and purchasing media from any other FluidMedia system. Systems can be located on a portable game player, a wireless connection in a nightclub, or a more traditional commercial web site. This maximizes the opportunity to acquire interesting media. For example, one could immediately purchase an interesting game from another portable game player to participate in an on-going networked game, or purchase an interesting song playing at a café or nightclub. Next, the system supports paying "kickbacks" to a user selling media to another user. While part of the purchase price is set aside to be eventually redeemed to the distributor of the media, the remaining part is made available to the seller for future purchases.

Typical sales transactions can be completed quickly, thus each user needs only sell a new file to a couple of other users for a spontaneous distribution tree to evolve and new content to propagate rapidly through a community of users. The ability for media to rapidly propagate through a community also means that the opportunities to sell media files diminish rapidly within a community. As a result, users that are motivated to sell media must act early or see their community saturated, providing further incentive for aggressive sales.

3 The FluidMedia system

The FluidMedia system we have built consists of several parts. HP iPAQ[1] Pocket PC's are used to model portable FluidMedia players. The iPAQ's are equipped with smart card readers, 802.11b wireless network cards and IR transceivers. Users connect to each other's iPAQ's using a combination of the IR transceivers to perform initial device discovery and an 802.11 wireless network configured in ad-hoc mode[3]. The software modules running on an iPAQ include a HTML Browser, a FluidMedia Server and the smart card (see Fig 1.) An HTTP connection is established between each of the iPAQ's and its opposing FluidMedia server, which then downloads a HTML description of the media files available on the device. Users then browse each other's media lists (currently limited to MP3 music files or MPEG2 video files) and can select media to purchase from each other. As presentation of available media files is independent of the transactions required to purchase the file, the user interface can be personalized by changing its appearance. Thus, for example, one can change the background of the page or even order songs in the order in which they have most recently been played, highlighting the currently playing song.

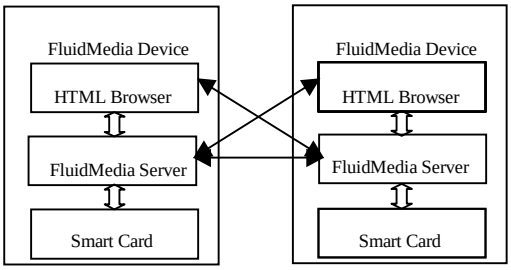

Fig 1. FluidMedia Device Architecture

Completing a purchase transaction is a three-step process. First, the smart cards authenticate each other and establish an encrypted connection over the network. Next, the smart cards verify that the user initiating the purchase has sufficient tokens to purchase the selected media. Assuming sufficient tokens, the purchasing smart card deducts the purchase price and the selling smart card delivers a media key to allow the purchasing iPAQ to decode the purchased file. During the transaction, the smart card logs some details of the transaction, such as the IDs of the peers doing the transaction and the fee to be paid to the distributor of the media.

The purchaser then has the option to download the actual media file. Before the transfer, the media file is decrypted and re-encrypted using a key suitable for the buying device, ensuring that it can only be decoded when a smart card with the right media key is present in the buying device. Once download is complete, the user can play the newly purchased file on the iPAQ.

Alternatively, a user can use an iPAQ to connect to a PC running a FluidMedia server. While the PC resident

[1] www.hp.com

service can perform the same media sale function as any other FluidMedia device, it is additionally capable of selling tokens to a user's smart card. Prior to updating the tokens on a smart card, the service deducts tokens representing fees due the media distributor and checks transaction logs from the smart card to detect possible fraudulent use of the system.

4 Security properties

4.1 Trust model

Since FluidMedia devices are used to collect royalty for the media provider and incentives for the owners, both of them have to make sure the devices function correctly. The media provider can achieve this by embedding a tamper-resistant module, such as a smart card, inside each FluidMedia device. The tamper-resistant module makes sure that only legal transactions are approved and credited. Meanwhile, the owners (i.e., users) also need to verify that they get paid for transactions happened under the control of the tamper-resistant module.

Chaum and Pedersen[4] describe a wallet whereby the owner of the wallet can monitor the communication between the tamper-resistant module and the outside world. However, their solution is expensive and geared toward protecting user privacy and not about verifying the internal behaviour of the tamper-resistant module. Although users are independent from the media provider, we believe in practice users can trust the media provider in the sense that they would let the media provider collect incentives on their behalf and make the incentives available to them though agreed process. Similar trust relationships exist in everyday lives, such as customer-bank relationship. Such a trust relationship between users and the media provider allows us to design a much simpler solution.

In our model, users trust the media provider to function on their behalf. Hence, our design is focused on how the media provider can defend against malicious users and third parties. As the only party to be protected, the media provider can implement a certified tamper-resistant module in each FluidMedia device while serving as the certificate authority for the system.

4.2 Security requirements

By breaking into the FluidMedia system, a hacker could potentially gain free media for himself and others. For example, a hacker may hack into a FluidMedia device to get a free copy of the media stored on the device or introduce fake tokens into the system to exchange for media from other devices. A hacker may also benefit other users at the cost of the media provider. FluidMedia system should defend against these attacks. Specifically, our FluidMedia system should satisfy the following security requirements.

- A FluidMedia player cannot be used to play pirated media or media that the user doesn't have rights to.
- Copyrighted media stored on a FluidMedia device cannot be easily pirated.
- It should be difficult to introduce fake tokens into the system so that the hacker is discouraged from doing it for personal gain. This means the money and knowledge investment needed to hack into FluidMedia system is greater than the gain in terms of free media.
- The FluidMedia system should be robust in the sense that hacking into a small number of devices will not bring down the whole system. The system should be able to identify hacked devices and exclude those devices from the system in a short period of time to limit the damage. The system should also prevent the hacker from benefiting other users in a manner that threatens the viability of the whole system.

5 Security design

5.1 Trusted module

FluidMedia system security is centered on a trusted module that is embedded in each FluidMedia device. This module is certified and trusted by the media provider. For the user, the trusted module is typically a tamper-resistant smart card issued by the media provider. The smart card is used to store tokens, digital rights and cryptographic keys. It is also used to perform certain cryptographic operations. Trusted modules can also be secure servers, e.g., trusted media servers run by the media provider.

FluidMedia smart cards are removable from the devices and can be used on any FluidMedia playing devices. From the user point of view, all sensitive data are stored in the smart cards and can be carried with the user. The FluidMedia playing devices are interchangeable and do not store any secrets.

All FluidMedia transactions are essentially happening between the smart cards. To facilitate mutual authentication and signing digital signatures, each smart card has a public-key pair certified by the media provider. The certificate is valid for a limited period of time and has to be periodically renewed. This renewal process allows the system to exclude fraudulent smart cards.

FluidMedia system uses tokens as local currency for media transactions. Tokens are issued by the media provider and stored as a counter inside the smart card.

Acquired digital rights are also stored in the smart card so that users can carry with them and use the rights spontaneously.

5.2 Media encryption

Component-based devices are much easier to hack than smart cards. For example, one could imagine the storage disk of a FluidMedia device being stripped out and scanned for playable media. Therefore, sensitive data (such as tokens, digital rights and cryptographic keys) are not stored on the devices. To discourage pirating, media stored on the devices are encrypted. The encryption of media is done in a per-copy per-device manner such that not only different songs are encrypted differently but also copies of the same song are encrypted differently on different devices. This means a hacker who obtains the decryption key for a song on one device cannot benefit others by publishing the key.

In FluidMedia, each device has a unique publicly known device ID D_d and each piece of copyrighted media has a unique publicly known media ID M_m. The media provider has a secret master key derivation key KP that is never given to anyone else. For each piece of copyrighted media M_m, the media provider computes the media key K_m using the following method[5]:

$$K_m = h(KP, M_m, KP) \qquad (1)$$

where h() is a secure hash function.

The actual media encryption/decryption key $K_{d,m}$ for media M_m on device D_d is then derived by:

$$K_{d,m} = h(K_m, D_d, K_m) \qquad (2)$$

The media key K_m essentially represents the digital right to media M_m and should only be given to those smart cards that have purchased media M_m. Note that access to K_m allows computing the media encryption key $K_{d,m}$ for any device D_d using (2). Hence, K_m should be stored in the smart cards and never exposed to the devices.

When playing media M_m, device D_d sends M_m and D_d to the smart card and asks for the decryption key $K_{d,m}$. The smart card reads key K_m corresponding to M_m from its memory, computes $K_{d,m}$ using (2) and returns $K_{d,m}$ to the device. A property of FluidMedia system is that the digital rights (i.e., K_m) for media files are stored inside the smart cards and independently of the devices. A user can carry his smart card with him and use the smart card to play entitled media on any FluidMedia devices.

During a transaction for media M_m, K_m is sent from the seller's smart card to the buyer's in a secure way. The seller can also derive the media encryption key $K_{d,m}$ of the buying device using the buyer's device ID obtained during the authentication process, encrypt the media itself using $K_{d,m}$ and transfer the encrypted media to the buyer. However, this step is not required. A feature of FluidMedia is the separation of media storage from digital rights management. Transaction of digital rights can be made separately from downloading of the media. Encrypted media can even be preloaded onto the devices.

5.3 Mutual authentication

Since FluidMedia devices only do media transactions between themselves, they need to authenticate each other. We use public-key cryptography for mutual authentication between two trusted modules. All trusted modules have public-key certificates issued by the media provider to facilitate mutual authentication and key exchange. The media provider serves as the certification authority (CA) for the FluidMedia system. The authentication protocol also derives a shared session key to be used for the protection of sensitive transaction data.

Suppose FluidMedia players (i.e., their respective smart cards) A and B having public-key pairs K_A and K_B, respectively. They also have the corresponding certificate C_A and C_B issued by the media provider MP. Both A and B have MP's public key so that they can verify certificates signed by MP. They agree on a secure hash function h(), such as SHA-1. The following protocol is used for mutual authentication between the two players (smart cards) and generating a shared session key K.

1. $A \rightarrow B$: C_A, $h(N_A)$, where N_A is a nonce chosen by A
2. $B \rightarrow A$: C_B, $E_{K_A}(N_B)$, $Sig[h(C_A, C_B, h(N_A), E_{K_A}(N_B))]$, where N_B is a nonce chosen by B, $E_X(Y)$ denotes the encryption of Y using public key X, Sig[X] denotes the digital signature on X
3. $A \rightarrow B$: $E_{K_B}(N_A)$, $Sig[h(C_A, C_B, E_{K_B}(N_A), E_{K_A}(N_B))]$

When successful, both A and B compute the session key $K = h(N_B, N_A, N_B)$.

In step 1, player A randomly chooses a number N_A and send $h(N_A)$ as its commitment to player B, together with its public key certificate C_A. N_A is chosen randomly and differently for each session and serves as both a challenge and A's contribution to the resulting session key K. Sending $h(N_A)$ rather than N_A in step 1 prevents player B from being able to determine the resulting session K on its own.

In step 2, player B verifies C_A using MP's public key. When the verification fails, it aborts the protocol. When successful, player B extracts player A's identity and public key K_A from C_A. Player B then randomly

choose a number N_B, encrypt it using K_A and sends it to player A together with a signature on all the values exchanged so far. The digital signature from player B proves to player A that B knows the private key corresponding to C_B, and therefore authenticates B to A. N_B should be chosen randomly and differently for each session since it serves as a challenge to A. N_B also is B's contribution toward the resulting session key K.

In step 3, player A verifies C_B and B's signature. If any of the verification fails, A aborts the protocol. Otherwise, A encrypts N_A using B's public key and sends it to B together with its signature on previously exchanged values. The signature serves as its response to B's challenge and authenticates itself to B. A then decrypts $E_{K_A}(N_B)$ to get N_B and computes session key $K = h(N_B, N_A, N_B)$.

After player B gets A's message sent in step 3, B verifies A's signature. If the verification fails, B aborts the protocol. If successful, B decrypts $E_{K_B}(N_A)$ to obtain M and verifies that h(M) is equal to the value $h(N_A)$ B received in step 1. If the verification fails, it aborts the protocol. If successful, B is satisfied that A didn't cheat and M is the value N_A that A committed in step 1. B then computes the same session key $K = h(N_B, N_A, N_B)$.

5.4 Fraudulent smart card exclusion

Smart cards are difficult to hack due to their packaging, but they are not immune to attacks[6]. In the case of hacked smart cards, a robust system should be able to detect those fraudulent cards and exclude them from the system in a timely fashion. FluidMedia system achieves robustness through examining transaction logs and periodic renewal of certificates.

For every FluidMedia transaction, a transaction record is added to the transaction log of each involved smart card. The transaction record includes the identities of both smart cards, time and price of the transaction, and digital signatures from each smart card. The transaction log is kept inside the smart card and is automatically uploaded to the media provider whenever a connection is established.

Certificates issued to a smart card by the media provider are only valid for a limited period, e.g. one month. Without a valid certificate, the smart card can still be used to play purchased media, but cannot perform new transactions, since a valid certificate is necessary for mutual authentication. Therefore, all smart cards must renew their certificates periodically in order to stay in the system.

The media provider can verify the authenticity of transaction logs by checking the digital signatures on the records. A fraudulent smart card cannot fake a transaction record by itself since it cannot generate a valid digital signature of its transaction partner. At most it can delete the transaction log. If the transaction partner is honest and reports the transaction, the media provider can detect such behavior and identify the fraudulent smart card. Hence, fraudulent smart cards can only fake records of transactions that happened between themselves. They cannot affect the functioning of honest smart cards in the system.

With authenticated transaction logs, the media provider can identify fraudulent smart cards and exclude them from the system by refusing to issue new certificates. When a smart card is hacked, the hacker can benefit himself and hence damage the system by introducing fake tokens and buy media for free. However, such behavior can be easily detected since fraudulent smart cards spend many more tokens than they make. The only way to escape detection is to restrict transactions to those between colluding fraudulent smart cards. In such cases, they can simultaneously delete transaction records and make those transactions invisible to the media provider. However, the benefit of doing so is very limited; it is equivalent to hacking into multiple smart cards and getting the digital rights stored on them for free. Given the tamper-resistance of smart cards, this is hardly worthwhile.

The window of opportunity to make fraudulent transactions before being excluded depends on the renewing period. The shorter the period the less benefit a hacker can get by hacking into a FluidMedia smart card. A compromise between ease of use and security is needed. We feel this is a small price to pay to get a simple and yet robust system.

6 Related work

The explosion of digital music on the Internet has put digital rights management (DRM) technology into the spotlight. DRM systems grants owners of digital content the ability to specify rules for that content. IBM introduced *Cryptolopes*[10] which are essentially objects containing a Java applet to interact with users, business rules for content usage, and an encrypted version of the content. When users want to access the content, they first pay for the content using the program and obtain a decryption key from a cryptolope clearing center located remotely. They then use a "trusted viewer" to view the content. InterTrust offers a similar product in *DigiBoxes*[11]. The approach adopted by Cryptolopes and DigiBoxes however has the following drawbacks: (1) it doesn't support offline peer-to-peer transactions; (2) its software-only approach doesn't provide adequate security.

We think persistent protection is only practical when secure hardware is used in the user's computer or appliance. The analysis of hardware requires specialized tools and skills. Unlike software, hardware has the ability to actively detect and respond to attacks. Secure hardware also allows us to integrate media transfer with payment. We choose to use smart cards as

the secure hardware for FluidMedia because it is ideally suited for carrying digital rights for the nomadic user.

Horne et al.[12] recognized quality assurance issues in peer-to-peer file sharing systems and piracy concerns for copyright holders, and introduced a system that uses economic incentives to motivate users to keep the content within the subscription community. However, their threat model is drastically different from ours. In their system, each user takes care of him/herself and there is no system-wide trusted hardware for secure payments and content quality control. Hence, the main problem they are trying to address is fair exchange of media and payment between mutually suspicious users. Their solution is to use an escrow server as a trusted third party for managing the risk.

7 Future work

Although media stored on a FluidMedia device are encrypted, when playing back stored media, a FluidMedia device will get the decryption key from the inserted smart card. Therefore, the software running on the device needs to be verified and trusted since it will know both the decryption key and the decrypted media.

Several existing secure bootstrap approaches look promising. For example, Tygar and Yee[7] described a model where a secure coprocessor is used for secure bootstrapping and the coprocessor itself is in turn authenticated by user smart cards. Arbaugh et al.[8] introduced another secure bootstrap architecture based on a small trusted portion of system ROM (BIOS), which fits well with existing operating systems.

We plan to further investigate and implement a secure bootstrap mechanism for FluidMedia devices that is verifiable using FluidMedia smart cards.

8 Conclusion

While we have dedicated considerable effort to developing a robust security mechanism, impenetrable security is not the most important aspect of the FluidMedia system. The genie is out of the bottle: the peer-to-peer media exchange systems are not going to go away. Nor is perfect copy protection likely achievable: in the end our human sensors are analog, and analog to digital conversion is getting better and more convenient every day[9]. Our emphasis has thus been to develop a reasonably secure system that offers significant flexibility and convenience for the typical user. We believe that people are willing to pay for such a system, and, in fact, will predominately choose a system that offers easy access to legally copied media.

9 References

[1] Janis Ian. The Internet Debacle – An Alternative View. http://www.janisian.com/article-internet_debacle.html

[2] Steve Petri. An Introduction to Smart Cards. http://www.sspsolutions.com/solutions/whitepapers/introduction_to_smart cards/?page=1

[3] Tourrilhes, Jean; Krishnan, Venky. Co-Link configuration: Using wireless diversity for more than just connectivity. HP Labs technical report, HPL-2002-258

[4] D. Chaum and T.P. Pedersen. Wallet databases with observers. In Advances in Cryptology: Crypto'92, volume 740 of Lecture Notes in Computer Science, pages 89-105, Springer, 1992.

[5] Uri Blumenthal, Nguyen C. Hien, and Bert Wijnen. Key Derivation for Network Management Applications," IEEE Network, pp.26-29, May/June 1997.

[6] Ross Anderson, Markus Kuhn, Tamper Resistance - a Cautionary Note, proceedings of the Second Usenix Workshop on Electronic Commerce, pp. 1--11, November 1996.

[7] J. D. Tygar and Bennet Yee. Dyad: A system for using physically secure coprocessors. Technical report, Carnegie Mellon University, May 1991. CMU-CS-91-140R.

[8] William A. Arbaugh, David J. Farber, and Jonathan M. Smith. A Secure and Reliable Bootstrap Architecture. In Proceedings 1997 IEEE Symposium on Security and Privacy, pages 65--71, May 1997.

[9] Charl Bergkamp. If it can be heard, it can be copied. http://www.computingsa.co.za/gifs/20010514offline.htm

[10] IBM infoMarket Development. IBM cryptolope containers. Tech. Report, IBM Corp., 1995.

[11] Olin Sibert, David Bernstein, and David Van Wie. Securing the content, not the wire, for information commerce. Tech. Report, InterTrust Technologies Corp., 1996.

[12] Bill Horne, Benny Pinkas and Tomas Sander. Escrow Services and Incentives in Peer-to-Peer Networks. In Proceedings of ACM conference on Electronic Commerce, October 2001.