A Payment Scheme for Broadcast Multimedia Streams*

Levente Buttyán and Naouel Ben Salem
Institute for Computer Communications and Applications
Communication Systems Department
Swiss Federal Institute of Technology – Lausanne
E-mail: Levente.Buttyan@epfl.ch, Naouel.BenSalem@epfl.ch

Abstract

Streaming multimedia data on the Internet is developing as a mainstream technology, which attracts many users by providing a new and convenient form of access to online multimedia information. While its strong business potential is obvious, many problems related to charging, copyright protection, and privacy can delay or even hinder its extensive deployment. In this paper, we are concerned with the charging problem, and propose an electronic payment scheme to use for purchasing broadcast multimedia streams. Our design respects the pay-per-use principle, makes cheating uninteresting for both the user and the service provider, resists against forgery and over-spending, protects sensitive payment information and user privacy, and allows the identification of misbehaving users.

1 Introduction

Streaming multimedia data on the Internet is developing as a mainstream technology [5], which attracts many users by providing a new and convenient form of access to online multimedia information. While its strong business potential is obvious, many problems related to charging, copyright protection, and privacy can delay or even hinder its extensive deployment. Therefore, it is important to fully understand these problems and find appropriate solutions.

In this paper, we are mainly concerned with the problem of charging, but we constantly keep the problems of user privacy and copyright protection in view as well. In particular, we present an electronic payment scheme to use for purchasing broadcast multimedia streams. The design of such a payment scheme should address many issues including the following:

- **Pay-per-use principle.** The payment system should allow the user to pay only for that part of the stream that she actually receives. This can be achieved (with some level of granularity) by using a micropayment system that supports very low value payments. A natural approach is to divide the stream into small pieces [8], and let the user pay for the pieces she receives using micropayments.
- **Fairness.** The payment system should guarantee fairness, meaning that either both the payer and the payee receive the expected item in the transaction (i.e., part of the stream paid for and the payment, respectively) or neither receives anything. Micropayment schemes proposed in the literature do not guarantee this property; fairness is sacrificed to the chief design goal, namely, efficiency. It is argued that the potential misbehavior of the parties is tolerable, since the potential loss is very low (the value of one micropayment token). While this is true considering one single transaction, it might be a problem considering the global system and longer time periods [2].
- **Robustness.** Clearly, the payment system should be robust and resistant to various attacks. In particular, forgery of money and over-spending should be prevented.
- **Protection of sensitive payment information.** Sensitive payment information should be well protected. People are often reluctant to use online services, because they are required to send sensitive payment information (e.g., their credit card numbers) to the service provider, and they fear that someone will learn and misuse this information. In fact, there are two dangers related to this issue: first, information can be eavesdropped by an attacker when it is transmitted from the user to the service provider via the network; and second, an attacker can break into the computer system of the service provider, on which user related payment information is stored, and obtain sensitive data. While the former can be prevented by using

standard security protocols, such as the Secure Socket Layer (SSL) [3], there is no standardized way to cope with the latter problem. There are, of course, secure databases in which sensitive information can be stored in a well protected way, but there is no guarantee whatsoever that the service provider uses such a database.

- **Protection of user privacy.** The payment system should not violate user privacy. Electronic payment systems often reveal some personal data of the payer to the payee. This may deter users from using such a system, because they fear that the service provider collects and misuses their personal information. Note that, in general, we cannot assume that the service provider is trusted by the user not to misuse her personal data, since it is likely that the service provider is completely unknown to the user.

- **Accountability.** It should be possible, however, to identify misbehaving users, notably, in order to support copyright protection. Therefore, a payment system that provides perfect user anonymity is not desirable.

Our payment scheme addresses all of these issues. In addition, it provides a solution to these problems in a rather simple way. The heart of the scheme is a robust micropayment protocol, which, although does not provide true fairness, does remove the financial incentive to cheat. We consider its achievement as a trade-off between efficiency and true fairness. Furthermore, in our scheme, sensitive payment data and personal user information is never revealed to the service provider, therefore, it cannot leak or misuse it. Instead, we introduce a special service provider, called the *broker*, which is competent in protecting sensitive information. On return of payment, the broker issues a *ticket* to the user, which allows her to access the service provided by the service provider in a semi-anonymous way. In addition, the ticket guarantees to the service provider that it will be paid for the service provision. The details are described in the forthcoming sections.

2 Overview

2.1 General model

We distinguish three main business roles: the *service provider*, who streams multimedia information via a broadcast channel (e.g., satellite, DVD, or Internet); the *user*, who receives the multimedia stream; and the *broker*, who issues tickets to the users that can be used to access the streams, and handles payments on behalf of users.

Since the stream is broadcast, it is received by potentially every user in the system. In order to ensure that only those users who paid for it can access the stream, we require that the stream is distributed in an encrypted form and users buy the appropriate keys to decrypt it. Therefore, conceptually, the service provider is divided into two components: a *content server* and a *key distribution server*.

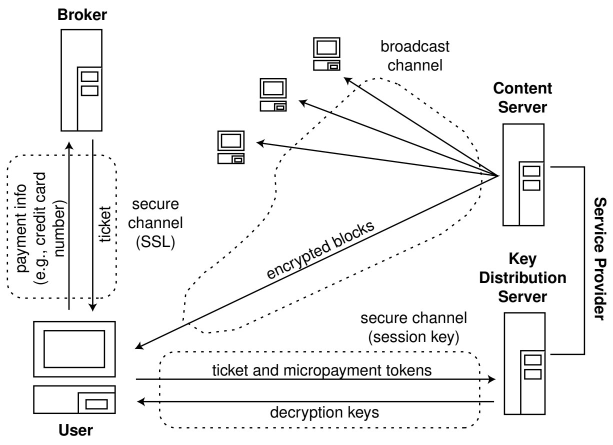

Figure 1. General model

The general model of our system is illustrated on Figure 1. The content server broadcasts the multimedia stream via a broadcast channel. The stream is organized into small blocks, each of which is encrypted with a different key. Potentially every user in the system can receive these encrypted blocks, but only those who buy the appropriate decryption keys can decrypt them.

In order to buy keys, the user first obtains a ticket from the broker. This transaction can be based on some electronic macropayment system, or it can simply use the traditional credit card system. In the latter case, the user sends her credit card number to the broker via a secure channel (e.g., using the SSL protocol). The broker is trusted to store credit card numbers and to handle payments on behalf of users, because it is specialized in this. This is similar to the trust in banks for storing money and handling payments in the physical world. The broker checks the solvency of the user and then returns a ticket.

The ticket contains data to initialize a micropayment session between the user and the key distribution server of the service provider. Conceptually, one might think of a ticket as a set of micropayment tokens, the total value of which is equal to the amount paid to the broker when the ticket was obtained. The user can spend these tokens to buy keys from the key distribution server.

Later, the service provider redeems the received micropayment tokens for real money at the broker (not shown on Figure 1). The broker pays using the money it collected from the user (e.g., it debits the credit card account of the user).

We should note that the ticket does not contain identity and payment information of the user, thus the user can safely use it to access the service. On the other hand, it contains a digital signature of the broker, which ensures the service provider that it will be paid for the service provision.

2.2 The broker

The broker is paid for the service it provides by the service providers. An easy solution is the following: When the service provider redeems a ticket, the broker does not pay the full value of the ticket to the service provider, but it keeps a fixed percentage as a commission. This is reasonable, because the service providers are interested in the service provided by the broker, since it can significantly increase their turnover by providing a robust and secure way for users to pay. This is similar to the way how merchants cover the costs of credit card transactions today.

In principle, any organization can setup and run a broker. However, since the broker should be trusted to store sensitive information and to handle payments on behalf of users in a secure way, most probably those organizations that already have a good reputation or the necessary resources to build up a good reputation (e.g., banks, telecommunication network operators, etc.) can be successful in this business.

In a variation of our system, the user may be registered with a broker, in which case the broker opens an account for the user. Then the user only needs to authenticate herself to the broker when she wants to obtain tickets. The appropriate amount is charged to the account of the user, which is settled at regular times (e.g., the broker sends monthly bills to the user). Organizations that already maintain user accounts and perform billing (e.g., telecommunication network operators, energy providers, etc.) may be interested in this variation of the scheme, because they can easily become a broker: they already have all the necessary infrastructure that is required for billing and usually they also have a good reputation (i.e., they are trusted by the users).

2.3 Implementation approaches

There are many ways to implement our general model described above. One possible approach would be to implement the key distribution server on smart cards (or some similar tamper resistant hardware), which would be distributed to each user. The smart card could store or generate (starting from a secure seed) the decryption keys for an encrypted stream. The ticket could be used to initialize the smart card, which would then output a requested key on return of a valid micropayment token (similar to pay-TV systems).

Another approach is to implement the key distribution server as a set of replicated servers operated by the service provider that can be reached by the users via the Internet. In this approach, the ticket could be used to establish a secure channel (needed for the secure distribution of decryption keys) between the user and the service provider and to initialize the micropayment session. In both approaches, the encrypted stream itself could be distributed via satellite, DVD, or the Internet. The broker would always be contacted via the Internet.

In this paper, we will focus on the latter approach: we assume that the key distribution server and the content server are servers on the Internet. However, our design presented below can easily be adopted to the former approach as well.

3 The micropayment protocol

We begin the detailed description of our payment scheme by introducing the micropayment protocol that serves as the core of our system. Because of space limitations, we present the protocol briefly. A detailed description can be found in [2].

Like many other micropayment protocols proposed in the literature [1, 4, 6, 7], ours as well is based on the successive release of elements in a chain of cryptographic hash values. When the user U wants to buy some services from a service provider S, she generates a fresh chain of hash values $w'_0, w_1, w'_1, w_2, w'_2, \ldots, w_n, w'_n$ by picking w'_n at random and then computing $w_i = h(w'_i)$ and $w'_{i-1} = h(w_i)$ for $i = n, n-1, \ldots, 1$. Every two consecutive hash values w_i, w'_i form a micropayment token. Therefore, intuitively, each hash value can be thought of as a half-token. w'_0 is called the root of the hash chain and it is not a half-token itself. When the hash chain is generated, U sends w'_0 to a broker B together with some payment information (e.g., a credit card number). B generates a ticket, which includes w'_0 and the identifier of B, and it is digitally signed by B. U starts the micropayment session by sending this ticket to S.

In principle, the i-th micropayment from U to S has three steps. First U sends the pair $(w_i, 2i-1)$ to S (the first half-token), then S provides the i-th piece of the service to U, and finally U sends the pair $(w'_i, 2i)$ to S (the second half-token). Each half-token can be checked by S using the previously received half-token. However, a typical service session consists of a series of consecutive micropayments, and U can send the second half-token of the i-th payment $(w'_i, 2i)$ and the first half-token of the $(i+1)$-st payment $(w_{i+1}, 2i+1)$ in a single message. Furthermore, since w'_i can always be computed from w_{i+1} by S, only the second pair $(w_{i+1}, 2i+1)$ has to be sent. A typical service session, thus, looks like this:

$U \rightarrow S :\ w_1, 1$
$U \leftarrow S :$ *first piece of the service*

$U \to S: \; w_2, 3$
$U \leftarrow S:$ *second piece of the service*
...
$U \to S: \; w_l, 2l-1$
$U \leftarrow S:$ *last piece of the service*
$U \to S: \; w'_l, 2l$

If everything goes well, then S can present the ticket and the pair $(w'_l, 2l)$ to B, which pays S the amount corresponding to l tokens and charges U with the same amount. If something goes wrong and the last half-token is missing, then S can only present the ticket and the pair $(w_l, 2l-1)$ to B. In this case, B pays S the amount corresponding to $l-1$ tokens and charges U with the amount corresponding to l tokens (which leaves the broker with a surplus of the value of one token). The surplus of the broker is handled according to some policy (e.g., it can be distributed to charity purposes).

In our micropayment protocol, none of the parties gain any financial advantage by cheating. The reason is that the service provider can never earn more than the value of the services it provided, since the user authorizes the broker to pay the service provider after she has received the service (by sending the second half-token). Similarly, the user can never receive services that are worth more than that she is charged for, because she lets the broker charge her before receiving the service (when releasing the first half-token). Thus, cheating is uninteresting for any of the parties.

4 Ticket contents and ticket protocols

In this section, we describe how we embedded the micropayment protocol introduced in the previous section in the payment scheme for broadcast multimedia streams. We first discuss the contents of the ticket issued by the broker, and then present the protocols for ticket acquisition, usage, and redemption.

4.1 Ticket contents

The ticket contains the following fields:

- TICKETID is a unique (at least with respect to the service provider specified in the PROVIDERINFO field) identifier of the ticket.
- EXPTIME specifies an expiration time after which the ticket should not be used by the user and cannot be redeemed by the service provider.
- TOKENVALUE is the value of one micropayment token in the ticket.
- BROKERINFO contains at least the identifier of the broker that issued this ticket. In addition, it may also contain the public key certificate of the broker.
- PROVIDERINFO contains at least the identifier of the service provider for which this ticket is valid, and the IP addresses and port numbers of its content server and its key distribution server.
- SERVICEINFO contains information about the stream, for which this ticket is valid.
- TECHPARAMS contains parameters that are necessary to initialize the micropayment scheme (i.e., w'_0), and a symmetric session key k encrypted with the public key K_S (i.e., $\{k\}_{K_S}$) of the key distribution server specified in the PROVIDERINFO field. The session key k is used to provide a secure channel between the user and the key distribution server.
- BROKERSIG is a digital signature of the broker on the previous fields, which prevents illegitimate modification and forgery of the ticket.

4.2 Ticket acquisition

1. The user U selects a service provider S, and a service (stream) of S that she wants to access. S may advertise its streams on a Web site, so the user can make her selection using a Web browser. Other necessary information, such as the IP addresses and the port numbers of the content server and the key distribution server of S, and the price of one encrypted block of the stream can also be downloaded from this Web site.
2. U generates a hash chain with appropriate length according to the micropayment protocol presented in the previous section, and a symmetric session key k. She encrypts k with the public key K_S of the key distribution server of S, which is also obtained from the Web site of S. U generates a large random number to be used as the ticket identifier, chooses an expiration time, and sets the value of one micropayment token according to the price of one encrypted block of the desired stream.
3. Then U submits the data required to construct a ticket (partly obtained from the Web site of S, partly generated by U herself in the previous step) to the broker B together with some payment information (e.g., her credit card number).
4. B checks the received payment information and the solvency of the user. If it is not satisfied, then the protocol is aborted. Otherwise, B constructs a ticket using the received information, digitally signs it, and returns it to U. It also stores the ticket and the payment information in its internal database for future use. The communication between U and B is protected using standard security protocols (e.g., SSL).

5. U verifies the received ticket and stores it together with the corresponding hash chain and session key.

4.3 Ticket usage

1. The user U connects to the key distribution server of the service provider S (the necessary addressing information is obtained from the ticket), and sends the ticket to the server.

2. The key distribution server verifies:

   - that the ticket contains the identifier of S, and if this is not the case, it aborts the protocol;
   - the expiration time of the ticket, and if the ticket is expired, it aborts the protocol;
   - that the value of one micropayment token of the ticket corresponds to the price of one block of the requested stream, and if this is not the case, it aborts the protocol;
   - the signature of the broker, and if it is not correct, then it aborts the protocol;
   - that the ticket is not stored in its internal database, where it stores used tickets until their expiration time; if it finds this ticket, it aborts the protocol.

3. If all the verifications are successful, then the key distribution server stores the ticket in its internal database.

4. Then the server decrypts the session key k, and sends a confirmation message to U. This message is encrypted with k.

5. U decrypts the received message. If this operation is successful, then U knows that the server successfully decrypted the session key, and there is now a secure channel between U and the server.

6. U sends the first half of the next micropayment token to the server together with the identifier of the block for which she wants to obtain the decryption key.

7. The server verifies the token, and sends back the requested decryption key (encrypted with the session key k) to U. It stores the last received token for later use.

8. The previous two steps are repeated as many times as required.

9. Finally, when U does not need more keys, she sends the second half of the last micropayment token to the server. The server stores this for later use.

4.4 Ticket redemption

1. The service provider S authenticates itself to the broker B specified in the ticket, and sends the ticket identifier and the last received half token to B.

2. B looks up its database and retrieves the corresponding ticket. If this operation is not successful, then it aborts the protocol.

3. B verifies:

   - that the provider information in the ticket corresponds to S, and if this is not the case, it aborts the protocol;
   - that the ticket is not expired; if the ticket is expired, it aborts the protocol;
   - that successive applications of the hash function on the last half token received from S results in the root of the hash chain w'_0 stored in the ticket; from this computation, it knows the number l of tokens used;
   - that this ticket has not been redeemed before, and if this is not the case, it aborts the protocol.

4. If all the verifications are successful, then B marks the ticket as redeemed, pays the service provider, and charges the user according to the rules specified in the micropayment protocol.

5 Analysis

In this section, we briefly discuss how our scheme addresses the problems listed in Section 1.

- **Pay-per-use principle.** In our scheme, the stream is divided into blocks, each of which is encrypted with a different key. The user can decide freely which blocks she wants to decrypt and she needs to pay only for the decryption keys of those blocks.
- **Fairness.** Our micropayment protocol does not provide true fairness, because any of the parties may misbehave and bring the other party in a disadvantageous situation. However, the protocol is constructed in such a way that misbehavior is not financially advantageous for the misbehaving party (i.e., misbehavior is practically uninteresting).
- **Robustness.** Our scheme is robust and resists against the common attacks on electronic payment systems. First, forgery of micropayment tokens is made difficult by using a strong cryptographic hash function to generate them. This ensures that generating a valid token is computationally infeasible for anybody but the

user, even if all the previous tokens in the chain are known. Second, creating fake tickets would require the forgery of the broker's digital signature. Third, the ticket specifies the service provider for which it is valid. This makes over-spending easily detectable, because the ticket can only be used with a single service provider, which logs the first usage of it and detects any attempts to use it again. The ticket is kept in the log until its expiration time, after which it is not accepted anymore. We should note, however, that specifying the accepting service provider in the ticket makes our scheme less flexible than most of the micropayment systems in which the user can make payments to any service provider without contacting the broker before the payment session.

- **Protection of sensitive payment information.** Sensitive payment information (e.g., the credit card number of the user) is not revealed to the service providers, but stored by the broker, which has the necessary competency to protect it. Furthermore, sensitive information is stored at a single place, instead of being stored by all the service providers that the user is in contact with, which would obviously increase the probability of theft. When transmitted to the broker, the payment information is protected using standard security protocols, such as SSL.

- **Protection of user privacy.** The ticket does not contain the identity of the user, thus, the service provider does not know who uses the service. The broker knows which services are used by the user, but we assume that the user trusts the broker not to misuse this knowledge. This trust is based on the good reputation of the broker.

- **Accountability.** The broker can always link a real user identity (or at least a valid credit card number) to a ticket, thus, a misbehaving user can be identified with the help of the broker, if this is deemed necessary.

6 Prototype implementation

We implemented the proposed payment scheme using the Java programming language. In our implementation, the service provider streams audio files to the users via the Internet. For streaming, we used the Java Media Framework package (www.javasoft.com).

As described in the scheme, the stream is divided into small blocks, each of which is encrypted with a different key. For the encryption of the blocks and other cryptographic operations of the micropayment protocol, we used the ABA (www.aba.net.au) implementation of the Java Cryptography Extension interface (www.javasoft.com).

We fully implemented only the ticket usage protocol. Ticket acquisition is simulated by a simple utility program that generates tickets from data stored in an input file. We did not implement the ticket redemption protocol.

7 Conclusion

We presented an electronic payment scheme for broadcast multimedia streams. Our scheme respects the pay-per-use principle, makes cheating uninteresting for both the user and the service provider, resists forgery and over-spending, protects sensitive payment information and user privacy, and allows the identification of misbehaving users. It uses a trusted party, called the broker, that issues a ticket to the user, which allows her to access the service provided by the service provider in a semi-anonymous way. In addition, the ticket guarantees to the service provider that it will be paid for the service provision. We described the protocols run by the user, the service provider, and the broker in details. We briefly analyzed our scheme and reported about a prototype implementation.

References

[1] R. Anderson, C. Manifavas, and C. Sutherland. NetCard – A practical electronic cash system. Technical Report, Cambridge University, Computer Laboratory, 1995.

[2] L. Buttyán. Removing the financial incentive to cheat in micropayment schemes. *IEE Electronics Letters*, 36(2):132–133, January 2000.

[3] A. Freier, P. Karlton, and P. Kocher. The SSL protocol. Version 3.0. Internet Draft. (available at http://home.netscape.com/eng/ssl3/)

[4] R. Hauser, M. Steiner, and M. Waidner. Micropayments based on iKP. Technical Report RZ 2791, IBM Research, February 1996.

[5] G. Lawton. Video streams into the mainstream. *IEEE Computer*, 33(7):12–17, July 2000.

[6] T. Pedersen. Electronic payment of small amounts. Technical Report DAIMI-PB-495, Aarhus University, Computer Science Department, 1995.

[7] R. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. Technical report, MIT Laboratory for Computer Science, 1996.

[8] E. von Faber, R. Hammelrath, and F-P. Heider. The secure distribution of digital contents. In *Proceedings of the 13th Annual Computer Security Applications Conference*, San Diego, California, December 1997.