Large-Scale Pay-As-You-Watch for Unicast and Multicast Communications *

Antoni Martínez-Ballesté, Francesc Sebé, Josep Domingo-Ferrer

Dept. of Computer Engineering and Maths,
Universitat Rovira i Virgili,
Av. Països Catalans 26,
E-43007 Tarragona, Catalonia, Spain
e-mail {anmartin, fsebe, jdomingo}@etse.urv.es

Abstract. This paper addresses the problem of pay-as-you-watch services over unicast and multicast communications. For each communication model, we present two solutions, non-verifiable and verifiable, depending on the existence or non-existence of trust between the source and the receiver(s). In verifiable schemes, the source obtains a proof of correct reception by the receiver(s); in non-verifiable schemes, receiver non-repudiation is not guaranteed, so there must be a trust relationship between source and receiver(s). While solutions for unicast pay-as-you-watch can be based on existing technologies, novel algorithms based on aggregation and multisignatures are needed and presented here to overcome implosion in multicast pay-as-you-watch.
Keywords: Pay-per-view systems, multicast, electronic payments.

1 Introduction

Several multimedia services consist of a customer or a set of customers who are interested in a certain content (say audio or video streams). In most of these services, the customer must nowadays pay for receiving or accessing the content. For instance, in current digital TV platforms, a flat monthly rate is paid to subscribe to a basic package of channels and services. Two different payment methods can be used to pay for the events not included in the flat rate:

– *Pay-per-view.* The content is viewed *after* the customer has paid. The customer pays for the whole piece of content. Thus, if she wants to stop watching anytime, she is losing a part of her money. This is the most common scheme used in current PayTV platforms for special events such as football matches, film premieres, etc.

* This work has been partly supported by the Spanish Ministry of Science and Technology and the European FEDER fund under project TIC2001-0633-C03-01 "STREAMOBILE".

– *Pay-as-you-watch.* Small payments are performed as contents are being streamed from the server to the customer. Pay-as-you-watch (or pay-as-you-listen) seems to be an option that fits better the customer needs. Successive payments can be performed every minute, for example. If a customer switches her player off, she has only paid for the minutes viewed so far.

1.1 Contribution and plan of this paper

Our paper deals with pay-as-you-watch, both for *unicast* (one-to-one communication) and *multicast* (one-to-n communication, usually with $n \gg 1$) transmission. For each communication model, we present two solutions, non-verifiable and verifiable, depending on the existence or non-existence of trust between the source and the receiver(s). In verifiable schemes, the source obtains a proof of correct reception by the receiver(s); in non-verifiable schemes, receiver non-repudiation is not guaranteed, so there must be a trust relationship between source and receiver(s). While solutions for unicast pay-as-you-watch can be based on existing technologies, novel algorithms based on aggregation and multisignatures are needed for multicast pay-as-you-watch in order to overcome the implosion problem resulting from many content receivers sending simultaneous payment to a content provider.

In Section 2, non-verifiable and verifiable pay-as-you-watch in a unicast scenario are discussed. Section 3 deals with non-verifiable and verifiable pay-as-you-watch problem in a multicast scenario. Finally, conclusions are summarized in Section 4.

2 Pay-as-you-watch in unicast communication

In unicast communication, the receiver directly establishes a session with the content provider (source). This is the distribution architecture currently used in most commercial pay-per-view services over the Internet. Unicast seems to be the most versatile option, because the subscriber can request a content at any time. Its main drawback is its lack of scalability on the server side: the server must have huge computing and communications resources to be able to service a large number of simultaneous unicast communications. Even with a large investment in hardware and bandwith, a peak in the number of unicast subscribers may result in service denial or degradation.

Some significant initiatives in unicast content distribution are CinemaNow [CinNw], Movielink [MovLn], Europe Online [EurOn] and

NDS [NDS], etc. Payment in those services is based on a combination of flat rate user subscription and advance pay per view for special events.

Let us consider a method for pay-as-you-watch in unicast communications. A first and easy solution for implementing a pay-as-you-watch service is to assume an agreement between the content provider and a telecommunications carrier; the buyer is then charged by the carrier for the time she has been enjoying the service and the carrier transfers payment to the content provider. Due to the universal and free access inherent to Internet, the aforementioned carrier-provider scheme does no longer work. On the Internet, there is no carrier (or there are many of them), so contents should be paid directly to the provider as they are being received.

2.1 Non-verifiable unicast solution

A non-verifiable pay-as-you-watch solution is one in which the provider obtains no proof of correct content reception by customers. Even without such a proof, a unicast provider is aware of the content received by each customer. The reason is that the content is sent using individual connections. Therefore, the provider can charge the customer for the minutes she has received. Thus, *the non-verifiable unicast solution consists of the provider metering the contents sent to each customer and thereafter billing the customer accordingly.*

The main drawback of non-verifiable systems is the need for trust between provider and customers:

- On one hand, the customer must trust the service provider: the customer must believe that the service provider will not charge her for contents she has not received.
- On the other hand, the provider cannot prove a subscriber is receiving a certain content. In this way, a dishonest customer could repudiate having received a certain content. After repudiation, the dishonest user could claim her money back and/or redistribute the received content without being punished.

2.2 Verifiable unicast solution

Since the number of simultaneous unicast customers is limited by the provider's outgoing bandwidth, it is highly unlikely that the number of customers is so large that the provider is swamped by the incoming flow of customer payments. Therefore, a pay-as-you-watch scheme can be put in place where the customer pays in real-time as she is receiving the content.

The transaction costs of standard electronic payments are usually considered too high for small amounts such as those required for real-time payment of small time slots. These transaction costs can be split into communication and computation costs, the latter being caused by the use of complex cryptographic techniques such as digital signatures. Micropayments are electronic payment methods specifically designed to keep transaction costs very low. In most micropayment systems in the literature, computational costs are dramatically reduced by replacing digital signatures with hash functions. This is the case of PayWord and Micromint[Riv95], where the security of coin minting rests on one-way hash functions.

In [Dom02], we proposed a verifiable pay-as-you-watch scheme based on PayWord. A prototype system offering that service is available at [Str04]. The operation of that system can be sketched as:

1. The customer subscribes to the service.
2. The service provider certifies the customer key.
3. The customer generates a PayWord chain and signs the root.
4. As the content is being served to the customer, several payments are requested by the provider. Each payment requires each customer to send a coupon, *i.e.* a piece of the PayWord chain, to the service provider.
5. If the customer stops paying, the provider pauses sending the stream.

3 Pay-as-you-watch in a multicast scenario

Multicast content distribution [Mill99] is a solution to overcome the lack of scalability of unicast communications.

Multicast consists of one source sending the same content to a set of n receivers, where usually $n \gg 1$. The main goal of multicast is to prevent the source from having to send the same content once for each customer: the content is replicated and distributed over a multicast tree, formed by multicast routers or active network nodes. In order to organize the distribution of the content, multicast sessions are advertised and interested customers join the multicast group for the upcoming session.

This approach is less versatile than the aforementioned unicast distribution: a group of customers must watch or listen to the same content at the same time. Thus, multicast distribution is suitable for large scale live events or near-on-demand video services.

In the near future, most multimedia delivery services are likely to operate in multicast mode to send content over the Internet. Given

that a large audience is possible, collecting real-time payment (pay-as-you-watch) from all customers may result in a bottleneck at the source or payment collector. This bottleneck, known as the *implosion problem* [Qui01], arises in any communication from n parties to one party (in our case from the the multicast customers to the multicast provider).

3.1 Non-verifiable multicast solution

In a multicast scenario, the source is not aware of the identity of all receivers [Fen97]. Thus, in principle, multicast pay-as-you-watch faces an intrinsic problem, because the provider needs to know how long each customer has been receiving the content. In encrypted multicast communications [MSEC03], the content is encrypted under a symmetric session key known only to the set of registered receivers. When a customer registers to join the session, a rekeying procedure is performed so as to let newcomer learn the (new) session key. In the same way, a rekeying procedure is performed when a customer leaves the current session: a new decryption key must be generated so that it is only known to the remaining receivers.

When a customer is interested in registering to a multicast session, she must request the decryption key. In this way, the source/provider knows exactly the moment at which the customer starts receiving the content. On the other side, the customer can disconnect without notifying the source. Hence, in a subscription-based service, customers must periodically confirm that they stay connected. This many-to-one confirmation communication must be private and authenticated and, unfortunately, can lead to implosion problems at the source. To remedy this, a scheme for secure reverse transmission of bits from the leaves (receivers) to the root (source) of a multicast tree was presented in [Dom04] which avoids the implosion problem.

A protocol for many to one bit transmission By using the protocol in [Dom04], a set of u users, with $u \gg 1$, send a bit of information to the source. This protocol provides secrecy and authentication, by means of symmetric cryptography. In order to avoid implosion, bits are aggregated by intermediate nodes of the multicast tree as they are sent up to the source of the tree. This aggregation operation of data packets inside the network requires the support of the network infrastructure in terms of processing resources. Active networks [Psou99] allow information to be handled in the core nodes of the network.

The protocol can be summarized as follows:

Protocol 1 (Many-to-one bit transmission).

1. The source generates a set of $2u$ intervals $[I_i^{min}, I_i^{max}]$, for i=1 to $2u$, which are sent to the users.
2. Let $\boxplus$ be a homomorphic addition, *i.e.* if $\boxplus$ is performed on two encrypted values, the ciphertext corresponding to the addition of cleartexts is obtained. The parameters for using $\boxplus$ are multicast to users.
3. In order to collect one bit from every user:
   (a) The source multicasts a challenge message v, which is used by the users to choose values $s_u^0 \in [I_u^{min}, I_u^{max}]$ and $s_u^1 \in [I_{u+1}^{min}, I_{u+1}^{max}]$ representing, respectively, bit values 0 and 1.
   (b) User u sends to her parent router in the multicast tree (the one she gets the multicast stream from) a message M_u containing s_u^0 or s_u^1, depending on the value of the bit she wants to send.
   (c) Intermediate routers generate a message $M_{desc} = M_1 \boxplus \cdots \boxplus M_d$, where d is the number of child nodes of the router, and sends M_{desc} up to their parent router.
   (d) Finally, the source obtains an aggregated message M from which she is able to efficiently retrieve all values s sent by the users.

In our approach, if a user/customer sends any of her secret values s_u^0 and s_u^1, it means that she is still online. This is useful for the source/provider to learn that the customer must be billed till at least the moment of receiving the last s_u^i. However, note that the provider does not obtain any proof, *i.e.* she could not convince a third party, that the customer is correctly receiving the multicast content. So a scheme where keepalive bits are sent using Protocol 1 is non-verifiable.

3.2 Verifiable multicast solution

As mentioned before, the main barrier to using traditional micropayment schemes for fee collection in multicast environments is the implosion problem. Nevertheless, due to the increase in the computational power of processors and the advances in digital signatures techniques, it is no longer obvious that the computational cost of a digital signature is still unaffordable for micropayments [Mic02].

By using verifiable payment subscription, the source can prove that a certain customer has received a certain portion of content. On the other hand, a customer cannot deny having received the content (non-repudiation).

Our proposal is secure and scalable. It is based on the concept of multisignature [Bol03]. Multisignatures allow any subgroup of a group of entities to jointly sign a document in such a way that any verifier is convinced that each member of the subgroup participated in the signature.

Definition 1 (Computational Diffie-Hellman problem (CDH)). The CDH problem consists of finding $h = g^{\log_g u \cdot \log_g v}$ given three random elements $\{g, u, v\}$ of a group.

Definition 2 (Decisional Diffie-Hellman problem (DDH)). The DDH problem consists of deciding whether four elements $\{g, u, v, h\}$ in a group satisfy $\log_g u = \log_v h$.

Definition 3 (Gap-Diffie-Hellman group).
A Gap-Diffie-Hellman (GDH) group is one in which the CDH problem is hard but the DDH problem is easy.

In [Bol03], a multisignature scheme is proposed which can be built over any Gap-Diffie-Hellman (GDH) group. We next recall that scheme.

Protocol 2 (GDH multisignature).

1. *Key generation.* Let G be a GDH group and g a generator of G. In order to generate her public-private key pair, a customer U chooses a random positive integer x_u (her private key) and publishes $y_u = g^{x_u}$ (her public key).
2. *Signature computation.* In order to sign a message m, a customer U computes her signature as $sig_u = \mathcal{H}(m)^{x_u}$, where $\mathcal{H}$ is a one-way hash function.
3. *Signature verification.* This signature is verified by solving the DDH problem over the elements

   $$\{g, y_u, \mathcal{H}(m), sig_u\}$$

   If the answer to the DDH problem is yes, then the signature is accepted as valid.
4. *Multisignature computation.* Given a customer U_1 with a public-private key pair (y_{u_1}, x_{u_1}) and a customer U_2 with a public-private key pair (y_{u_2}, x_{u_2}), a multisignature of U_1 and U_2 on a message m is computed as follows:

(a) U_1 computes $sig_{u_1} = \mathcal{H}(m)^{u_1}$
(b) U_2 computes $sig_{u_2} = \mathcal{H}(m)^{u_2}$
(c) The multisignature on m is then computed as

$$sig_{u_1,u_2} = sig_{u_1} \cdot sig_{u_2}$$

5. *Multisignature verification.* The multisignature can be verified by solving the DDH problem over the elements

$$\{g, y_{u_1} \cdot y_{u_2}, \mathcal{H}(m), sig_{u_1,u_2}\}$$

If the answer to the DDH problem is yes, then the multisignature is accepted as valid.

The generalization of the above multisignature computation and verification to a set of n customers is straightforward. Next, we propose a scalable solution whereby the multicast source/provider can collect a proof that all customers registered in a multicast session have received a specific piece of content.

Protocol 3 (Aggregation using multisignatures).

1. *Customer registration.* In our proposal, we require each customer U_i to have a public-private key pair (y_{u_i}, x_{u_i}). The public key must be certified by a trusted certification authority.
2. *Payment request.*

(a) The source generates a message m specifying the content, the time slot and the amount of money to be paid. This message m is multicast to the set of registered users who are currently receiving the content.
(b) Upon reception, customers check the content of m for correctness and sign it. That is, customer U_i computes $sig_{u_i} = \mathcal{H}(m)^{u_i}$ and sends it up to her parent router in the multicast tree.
(c) Intermediate routers check the correctness of the received signatures, aggregate them by generating a multisignature on m and send the aggregated signature up to their parent router in the multicast tree.
(d) Upon reception of the final multisignature, the source checks its correctness.

Some remarks on Protocol 3 are in order:

– The final multisignature received by the source can be used to prove to a third party that a specific subset of customers signed the receipt and the payment corresponding to the content described in m. Thus Protocol 3 results in a verifiable solution for pay-as-you-watch multicast transmission.
– In case one of the customers leaves the group or fails to send a valid signature, a new rekeying process will be performed so that the failing customer is excluded from knowledge of the new session key.
– The size of the multisignature does not increase when aggregating signatures. This makes our proposal scalable, because the source will not be imploded by reception of a final aggregated signature which has the same size as the individual customer signatures.

4 Conclusions

We have presented in this paper several approaches to pay-as-you-watch transmission, both in unicast and multicast scenarios. For both scenarios, we have proposed non-verifiable and verifiable solutions. In a non-verifiable solution, the content provider does not obtain any proof that the customer has correctly received a specific piece of content; thus, a trust relationship between customers and provider is needed. Verifiable solutions are more versatile in that they can be implemented in the absence of trust.

In the case of unicast, existing technology can be adapted to produce both non-verifiable and verifiable solutions. Content metering and subsequent billing yields a non-verifiable scheme. Standard micropayments can be used to construct a verifiable scheme.

In the case of multicast, innovative protocols are required to overcome the implosion problem caused by a huge number of real-time micropayments being sent by customers to the multicast provider. The protocols we have proposed rely on aggregation of information at the intermediate multicast routers. If the aggregated information is not signed, we obtain a non-verifiable scheme; if it is signed, the resulting scheme is verifiable. In particular we have shown how to efficiently aggregate signed information using multisignatures.

References

[Bol03] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-Group signature scheme", in *Int. Workshop on Theory and Practice in Public Key Cryptography-PKC'2003*, LNCS 2567, Berlin: Springer-Verlag, pp. 31-46, 2003.

[CinNw] CinemaNow, http://www.cinemanow.com
[EurOn] Europe Online, http://www.europeonline.com
[Dom02] J. Domingo-Ferrer and A. Martínez-Ballesté, "STREAMOBILE: pay-per-view video streaming to mobile devices over the Internet", in *Proceedings of the 13th International Workshop on Database and Expert Systems and Applications (DEXA'2002)*, Los Alamitos CA: IEEE Computer Society, pp. 418-422, 2002.
[Dom04] J. Domingo-Ferrer, A. Martínez-Ballesté and F. Sebé, "Secure Reverse Communication in a Multicast Tree" in *Third International IFIP-TC6 Networking Conference - NETWORKING 2004*, LNCS 3042, Berlin: Springer-Verlag, pp. 807-816, 2004.
[Fen97] W. Fenner. Internet Group Management Protocol, Version 2. RFC 2236, November 1997.
[Mic02] S. Micali and R. L. Rivest, "Micropayments revisited" in *Topics in Cryptology - CT-RSA 2002*, LNCS 2271, Berlin: Springer-Verlag, pp. 149-163, 2002.
[Mill99] C. K. Miller, *Multicast Networking and Applications*. Reading MA: Addison Wesley, 1999.
[MovLn] MovieLink, http://www.movielink.com
[MSEC03] Multicast Security Working Group (MSEC WG). http://www.securemulticast.org
[NDS] NDS, http://www.nds.com
[Opp02] R. Oppliger, *Security Technologies for the World Wide Web (2nd Edition)*. Norwood MA: Artech House, 2002.
[Psou99] K. Psounis, "Active networks: Applications, security, safety and architectures", *IEEE Communication Surveys*, vol. 2, no. 1, pp. 1-16, 1999.
[Qui01] B. Quinn and K. Almeroth, "IP multicast applications: challenges and solutions", Internet RFC 3170, Sept. 2001. http://www.ietf.org
[Riv95] R. Rivest and A. Shamir, "PayWord and Micromint: Two simple micropayment schemes", Technical Report, MIT LCS, Nov. 1995.
[Str04] http://vneumann.etse.urv.es/streamobile