Securing Video Delivery to the Android Platform

Raimund K. Ege
Department of Computer Science
Northern Illinois University
DeKalb, IL 60115, USA
ege@niu.edu

ABSTRACT

The Android smartphone platform has gained significant market share among mobile devices with high-speed network connectivity. In this paper we describe Oghma, which is an Android application to make multimedia data, i.e. video, easily accessible. It uses a peer-to-peer content delivery model to ensure efficient delivery to a large audience. However, the openness of delivery demands adaptive and robust management of intellectual property rights. In this paper we describe a framework and its implementation to address the central issues in content delivery: a scalable peer-to-peer-based content delivery model, paired with a secure access control model that enables data providers to maintain control over their original content available. Our prototype implementation for the Android platform for mobile phones is described in detail.

Keywords: Android video client; peer-to-peer content delivery; mobile secure access.

1. INTRODUCTION

The Apple iPhone, and now increasingly Android-based smart phones, have ushered in a new era in omni-present broadband media consumption. Services such as YouTube, iTunes, Joost and Hulu have popularized delivery of audio and video content to anybody with a broadband Internet connection. High bandwidth internet connectivity is no longer limited to reaching PCs and laptops: a new generation of devices, such as netbooks and smart phones, is within reach of 3G/4G telecommunication networks.

In this paper we describe a new "app" for Android phones that delivers video in a secure and managed way. Figure 1 shows a screenshot of the Android home screen featuring our new Oghma secure multi-media delivery "app." Delivering multimedia services has many challenges: the ever increasing size of the data requires elaborate delivery networks to handle peek network traffic. Another challenge is to secure and protect the property rights of the media owners. A common approach to large-scale distribution is a peer-to-peer model, where clients that download data immediately become intermediates in a delivery chain to further clients. The dynamism of peer-to-peer communities means that principals who offer services will meet requests from unrelated or unknown peers. Peers need to collaborate and obtain services within an environment that is unfamiliar or even hostile.

Therefore, peers have to manage the risks involved in the collaboration when prior experience and knowledge about each other are incomplete. One way to address this uncertainty is to develop and establish trust among peers. Trust can be built by either a trusted third party [2] or by community-based feedback from past experiences [3] in a self-regulating system. Conventional approaches rely on well-defined access control models [4, 5] that qualify peers and determine authorization based on predefined permissions. In such a complex and collaborative world, a peer can benefit and protect itself only if it can respond to new peers and enforce access control by assigning proper privileges to new peers.

The broader goal of our work is to address the trust in peers which are allowed to participate in the content delivery process, to minimize the risk and to maximize the reward garnered from releasing data in to the network. In our prior work [1, 9] we focused on modeling the nature of risk and

Figure 1. Android Home Screen

reward when releasing content to the Internet. We integrated trust evaluation for usage control with an analysis of risk and reward. Underlying our framework is a formal computational model of trust and access control. In the work reported here we focus on the implementation aspects of the framework.

Our paper is organized as follows: the next section describes our prototype architecture that is based on a bittorrent-style of peer-to-peer content delivery. A central trust server tracks and manages peers and maintains a database of trust information. Peers can serve both as source and as consumer of data. Section III introduces our prototype client for the Android platform. Section IV elaborates on details of the Java implementation of the tracker, source and peer processes. Data is exchanged using the Stream Control Transmission Protocol (SCTP) which improves over the current standard-bearers Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) for multi-stream session-oriented delivery of large multi-media files over fast networks. Data is secured using a PKI-style exchange of public keys and data encryption. Section V elaborates on how a data provider and its peers can quantify gain from participating in the content delivery. It also explains our risk/reward model that enables a data source to initially decide on whether to share the content and keep some leverage after its release.

The paper concludes with our perspective on how modern content delivery approaches will usher in a new generation of Internet applications.

2. PROTOTYPE ARCHITECTURE

Peer-to-peer (P2P) delivery of multimedia aims to deliver multi-media content from a source to a large number of clients. For our framework, we assume that the content comes into existence at a source. A simple example of creating such multimedia might be a video clip taken with a camera and a microphone, or more likely video captured via a cell phone camera, and then transferred to the source. Likewise the client consumes the content, e.g. by displaying it on a computing device monitor, which again might be a smart phone screen watching a YouTube video. We further assume that there is just one original source, but that there are many clients that want to receive the data. The clients value their viewing experience, and our goal is to reward the source for making the video available.

In a P2P delivery approach, each client participates in the further delivery of the content. Each client makes part or all of the original content available to further clients. The clients become peers in a peer-to-peer delivery model. Such an approach is specifically geared towards being able to scale effortlessly to support millions of clients without much prior notice, i.e. be able to handle a "mob-like" behavior of clients.

The exact details of delivery may depend on the nature of the source data: for example, video data is made available at a preset quality using a variable-rate video encoder. The source data stream is divided into fixed length sequential frames: each frame is identified by its frame number. Clients request frames in sequence, receive the frame and reassemble the video stream which is then displayed using a suitable video decoder and display utility. The video stream is encoded in such a fashion that missing frames don't prevent a resulting video to be shown, but rather a video of lesser bit-rate encoding, i.e. quality, will result [7]. We explicitly allow the video stream to be quite malleable, i.e. the quality of delivery need not be constant and there is no harm if extra frames find their way into the stream. It is actually a key element of our approach that the stream can be enriched as part of the delivery process.

In our approach, multi-media sources are advertised and made available via a central tracking service: at first, this tracker only knows the network location of the source server. Clients that want to access the source do so via the tracker: they contact the tracker, which will respond with the location of the source. The tracker will also remember (or track) the clients as potential new sources of the data. Subsequent client requests to the tracker are answered with all known locations of sources: the original and the known clients. Clients that receive locations of sources from the tracker issue frame requests immediately to all sources. As the sources deliver frames to a client, the client stores them. The client then assumes a server role and also answers requests for frames that they have received already, which will enable a cascading effect, which establishes a P2P network where each client is a peer. Every client constantly monitors the rate of response it gets from the sources and adjusts its connections to the sources from which the highest throughput rate can be achieved.

Figure 2 shows an example snapshot of a content delivery network with one source, one tracker, 2 intermediate peers and one peer client. The source is where the video data is produced, encoded and made available. The tracker knows the network location of the source. Clients connect to the tracker first and then maintain sessions for the duration of the download: the 2 peers and the single client maintain an active connection to the tracker. The tracker informs the peers and client which source to download from: peer 1 is fed directly from the source; peer 2 joined somewhat later and is now being served from the source and peer 1; the client (peer 3) joined last and is being served from peer 1 and peer 2. In this example, peer 1 and 2 started out as clients, but became peers once they had enough data to start serving as intermediaries on the delivery path from original source to ultimate client.

We further enhance the basic bittorrent-style architecture by expanding the role of the tracker: in addition to managing which sources are made available to clients and peers, the tracker is elevated to a trust manager and clearing house. The tracker/trust manager controls access to the list of available servers.

3. ANDROID CLIENT

We chose the new and emerging Android platform to implement a proof-of-concept client for a mobile device. Android is part of the Open Handset Alliance [10]. Android uses a Linux kernel and internally uses many of the typical GNU Unix utilities. The graphical user interface is implemented in Java and therefore offers a flexible and standard set of communication and security features. It provides for both the Java Cryptography Architecture (JCA) as well as the Java Cryptography Extensions (JCE) and uses the popular and lightweight Bouncy Castle provider implementation [13].

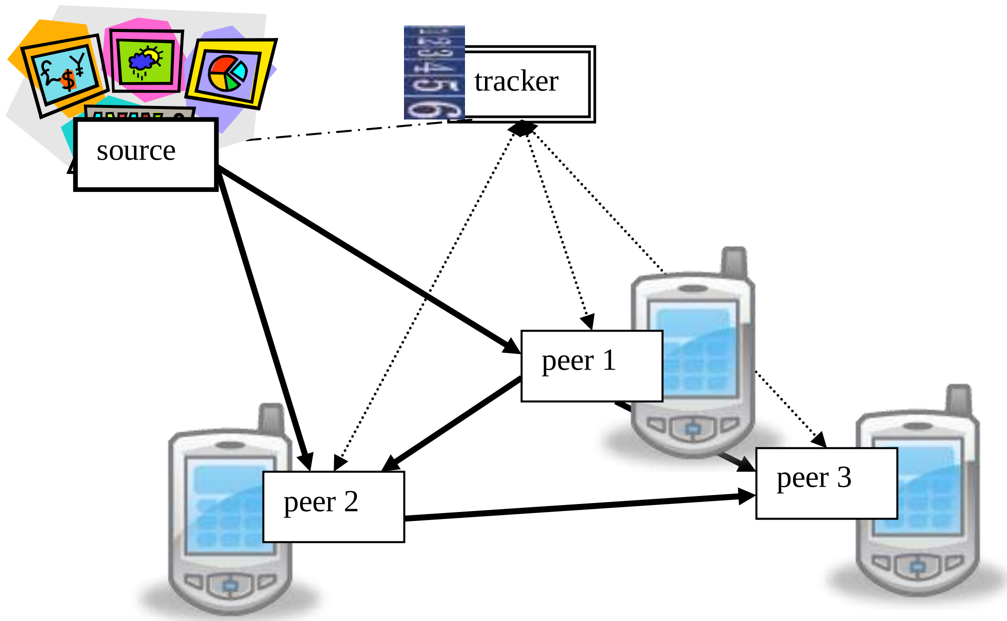

Figure 2. P2P Content Delivery Network

Figure 3. Oghma Login Screen

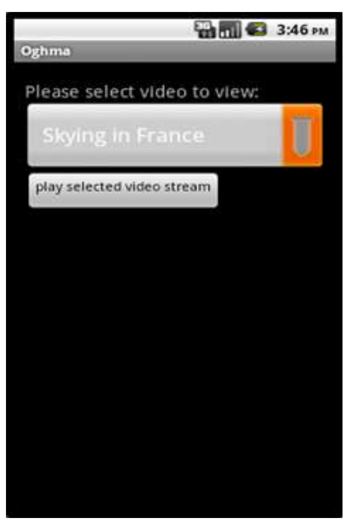

Figure 4. Stream Selection

Figure 5. Download Starting

Figures 3, 4, 5 and 6 show four sample screen shots taken from the Android system. It shows our Oghma Secure P2P media client. Figure 3 shows the login screen to our Oghma mobile client. It uses OpenID[6] user credentials and allows to establish a connection to a tracker URL. The user then activates the “Login” button, which starts the authentication process. The Oghma client establishes a connection with the trust server running at the specified URL. The trust server authenticates the client via its OpenID and password, and then transmits back a list of available video streams. The Oghma client switches to the screen shown in Figure 4: it shows the list of videos available from this source. After the user has made a selection, the screen shown in Figure 5 appears. The client starts downloading the video from the available video sources. The user is not aware that multiple network locations might be actually serving video fragments. Once a sufficient read-ahead buffer has been accumulated, the video stream starts playing on the Android device (Figure 6).

Figure 6. Oghma Video Display

4. IMPLEMENTATION DETAIL

Our implementation framework features 3 types of participants: (A) Trust manager, where all information on the current status of the content delivery network is maintained and all access decisions are made. (B) Source, where the data is available for further dissemination. The original source is the first source. Peers that have downloaded and consumed the data can become new sources; and (C) Client, where the consumption of the data occurs.

A. Trust Manager

The core of the content delivery model is the trust manager. It serves as the central clearing house for access decisions and also serves as the “tracker” who knows the location of the original/first source. It maintains a database of peer information: each peer is authenticated with an OpenID and carries historical data on past peer behavior.

Peers that wish to participate in the content delivery must first locate the tracker. A peer will start by establishing a connection to a tracker. Peers use their openID and password to login to the tracker. If the peer is known to the trust server, a list of available video segments and their sources is sent back to the peer. If the peer is new and not yet listed in the manager’s database, then – after all available data on the peer’s trustworthiness has been gathered – the process of creating a new database entry is initiated: the trust manager sends general key generation parameter specification detail to the peer, which uses it to generate a Diffie-Hellman pair of keys [11]. The peer then transmits its public key to the tracker. The trust manager completes the database entry with an initial trust value and informs the new client about which video segments are available for it. Figure 7 shows the trust manager’s graphical user interface. Figure 8 shows the security information, i.e. the public key, for the peer with openId “RaimundEge@gmail.com.”

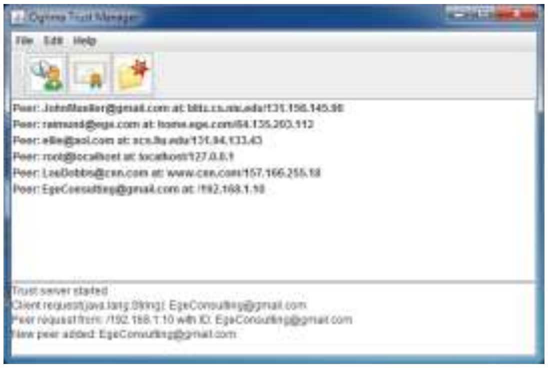

Figure 7. Oghma Trust Manager Peer database

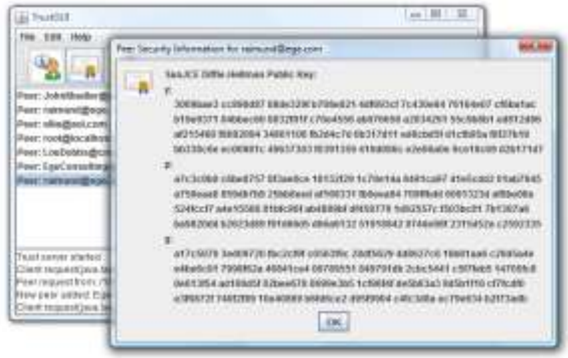

Figure 8. Security information

B. Source

At least one source must exist for the content delivery network to get started. The source first establishes contact with the trust manager/tracker. It also generates a Diffie-Hellman public/private key pair [11] and transmits its public key to the tracker. It then stands ready for data requests from clients. If a request from a client is received, it requests the client's public key from the tracker. With that public key and the source's own private key a Diffie-Hellman key agreement algorithm [12] is used to produce a session key. The session is then used by the source to encrypt all data that is sent to the client.

C. Client

The key to a smooth scaling of this ad-hoc p2p network is the algorithm used by the client to request frames from a source (either the original source or another client). A client consists of three processes:

1) a process to communicate with the tracker. The client and tracker to enable the decision on whether the peer is admitted into the content delivery network. The tracker informs the client which sources to use with their public keys. The client will update the tracker on its success in downloading the source data;
2) a process to request data from the given sources. Frames may be requested from multiple sources. Frames that are received are decrypted using a session key that is established via a key agreement using the public key of the data source and the client's own private key.
3) a process to sequence the frames/fragments received from sources and to assemble them into a usable media stream.

Our prototype uses the Java implementation of the SCTP [14] transport layer protocol. SCTP is serving in a similar role as the popular TCP and UDP protocols. It provides some of the same service features of both, ensuring reliable, in-sequence transport of messages with congestion control. We chose SCTP because of its ability to deliver multimedia in multiple streams. Once a client has established a SCTP association with a server, packages can be exchanged with high speed and low latency. Each association can support multiple streams, where the packages that are sent within one stream are guaranteed to arrive in sequence. Each source can divide the original video stream into set of streams meant to be displayed in an overlay fashion. Streams can be arranged in a way that the more streams are fully received by a client, the better the viewing quality will be. The first stream is used to deliver a basic low quality version of the video stream. The second and consecutive streams will carry frames that are overlaid onto the primary stream for the purpose of increasing the quality. In our framework we also use the additional streams to carry content that is "added value", such as advertising messages or identifying logos. The ultimate client that displays the content to a user will combine all streams into one viewing experience. The second feature of SCTP we use is its new class "SctpMultiChannel" which can establish a one-to-many association for a single server to multiple clients. The SctpMultiChannel is able to recognize which client is sending a request and enables that the response is sent to that exact same client. This is much more efficient than a traditional "server socket" which for each incoming request spawns a subprocess with its own socket to serve the client.

Our prototype implementation has three main components, which employ the three participant categories of our framework: The "Oghma Trust Manager" Java application implements the "trust manager" and tracker behavior. The "Oghma Source Main" Java application provides "source" capabilities only, whereas the "Oghma Client" Java application provides "client" and "source" capabilities. The implementation is composed from classes that implement the core behavior of maintaining communication sessions, accepting requests for frames and delivering them, and requesting and receiving frames. The major classes are FrameRequestor and FrameServer. The "Oghma Source Main" Java application starts out as the sole instance of FrameServer. The first client starts out as the sole instance of FrameRequestor. As the client accumulates frames it then also instantiates a FrameServer that is able to receive requests from other clients. A client that contains both a FrameRequestor and FrameServer instance becomes a true peer in the P2P content delivery framework.

5. UNITS of RISK and REWARD

We assume that the data made available at the source has value. Releasing the data to the Internet carries potential for reaping some of the value, but also carries the risk that the data will be consumed without rewarding the original source. There is also a cost associated with releasing the data, i.e. storage and transmission cost. For example, consider a typical "viral" video found on YouTube.com: the video is uploaded onto YouTube.com for free, stored and transmitted by YouTube.com and viewed by a large audience. The only entity that is getting rewarded is YouTube.com, which will accompany the video presentation with paid advertising. The person that took the video and transferred it to YouTube.com has no reward: the only benefit that the original source of the video gets is notoriety.

In order to provide a model or framework to asses risk and reward, we need to quantize aspects of the information interchange between the original source, the transmitting medium and the final consumer of the data. In a traditional fee for service model the reward *"R"* to the source is the fee *"F"* paid by the consumer minus the cost *"D"* of delivery:

$$R = F - D$$

The cost of delivery *"D"* consist of the storage cost at the server, and the cost of feeding it into the Internet. In the case of YouTube, considerable cost is incurred for providing the necessary server network and their bandwidth to the Internet. YouTube recovers that cost by adding paid advertising on the source web page as well as adding paid advertising onto the video stream. YouTube's business model recognizes that these paid advertisings represent significant added value. As soon as we recognize that the value gained is not an insignificant amount, the focus of the formula shifts from providing value to the original data source to the reward that can be gained by the transmitter. If we quantify the advertising reward as *"A"* the formula now becomes:

$$R = F - (D - A)$$

Even in this simplest form, we recognize that *"A"* has the potential to outweigh *"D"* and therefore reduce the need for *"F"*. As YouTube recognizes, the reward lies in *"A"*, which is paid advertising that accompanies the video.

In some of our prior work [8] we focused on mediation frameworks that capture the mutative nature of data delivery on the Internet. As data travels from a source to a client on lengthy path, each node in the path may act as mediator. A mediator transforms data from an input perspective to an output perspective. In the simplest scenario, the data that is fed into the delivery network by the source and is received by the ultimate client unchanged: i.e. each mediator just passes its input data along as output data. However, that is not the necessary scenario anymore: the great variety of client devices already necessitate that the data is transformed to enhance the client's viewing experience. We apply this mediation approach to each peer on the path from source to client. Each peer may serve as a mediator that transforms the content stream in some fashion. Our implementation employs the stream control transmission protocol (SCTP) which allows multi-media to be delivered in multiple concurrent streams. All a peer needs to do is add an additional stream for a video overlay message to the content as it passes through.

The formula for reward can now be extended into the P2P content delivery domain, where a large number of peers serve as the transmission/storage medium. Assuming "*n*" number of peers that participate and potentially add value the formula for the reward per peer is now:

$$R_p = \sum_{i=1}^{n} (F_i - (D_i - A_i)) - F_p$$

D_i and A_i are now the delivery cost and value incurred at each peer that participates in the P2P content delivery. F_i is the fee potentially paid by each peer. F_p is the fee paid to the data source provider. Whether or not the data originator will gain any reward depends on whether the client and the peers are willing to share their gain from the added value. In a scenario where clients and peers are authenticated and the release of the data is predicated by a contractual agreement, the source will reap the complete benefit.

In our model we quantify the certainty of whether the client and peers will remit their gain to the source with a value of trust *"T"*: *T* represents the trust in the client that consume that data, *T* represents the trust in each peer that participates in the content delivery. The trust is evaluated based on both actual observations and recommendations from referees. Observations are based on previous interactions with the peer. Recommendations may include signed trust-assertions from other principals, or a list of referees that can be contacted for recommendations. The trust value, calculated from observations and recommendations, is a value within the [0, 1] interval evaluated for each peer that requests to be part of the content delivery.

Our model enables an informed decision on whether to accept a new peer based on the potential additional reward gained correlated to the risk/trust encumbered by the new peer.

6. CONCLUSION

In this paper we have described a model and framework for a new generation of content delivery networks. We have described a prototype implementation that follows a bittorrent-style of P2P network, where a tracker/trust manager controls access to the content delivery network and disseminates information on which sources are available to download from. Our implementation includes a Java-based client for the Android platform for smart phones. Such P2P content delivery has great potential to enable large scale delivery of multimedia content. Our framework is designed to enable content originators to assess the potential reward from distributing the content to the Internet. The reward is quantified as the value added at each peer in the content delivery network and gauged relative to the actual cost incurred in data delivery but also correlated to the risk that such open delivery poses.

Consider the scenario we described earlier in the paper: a typical "viral" video found on YouTube.com: the video is uploaded onto YouTube.com for free, stored and transmitted by YouTube.com and viewed by a large audience. The only entity that is getting a reward is YouTube.com, which will accompany the video presentation with paid advertising. The only benefit that the original source of the video gets is notoriety. Using our model, the original data owner can select other venues to make the video available via a peer-to-peer approach. The selection on who will participate can be based on how much each peer contributes in terms of reward but also risk. Peers will have an interest in being part of the delivery network, much like YouTube.com has recognized its value. Peers might even add their own value to the delivery and share the proceeds with the original source.

Whereas in the YouTube.com approach the reward is only reaped by one, and the original source has shouldered all the risk, i.e. lost all reward from the content, our model will enable a more equitable mechanism for sharing the cost and reward. Our model might just enable truly open content delivery via the Internet.

7. REFERENCES

[1] Raimund K. Ege, Li Yang, Richard Whittaker. Extracting Value from P2P Content Delivery. 4th Intl. Conf. on Systems (ICONS 09), p.102-108, Cancun, Mexico, 2009.

[2] Y. Atif. Building trust in E-commerce. IEEE Internet Computing, 6(1):18–24, 2002.

[3] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. Comm. of the ACM, 43(12):45–48, 2000.

[4] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca. A logical framework for reasoning about access control models. 6th ACM Symp. on Access control models & technologies, p. 41–52, New York, 2001.

[5] S. Jajodia, et al. Flexible support for multiple access control policies. ACM Transaction Database System, 26(2):214–260, 2001.

[6] OpenID, http://openid.net/add-openid/ [accessed 10-5-10]

[7] C. Wu, Baochun Li. R-Stream: Resilient peer-to-peer streaming with rateless codes. 13th ACM International Conference on Multimedia, p. 307-310, Singapore, 2005.

[8] R. K. Ege, L. Yang, Q. Kharma, and X. Ni. Three-layered mediator architecture based on dht. Proceedings of the 7th Intl. Symp. on Parallel Architectures, Algorithms, and Networks (I-SPAN 2004), Hong Kong, China, 2004.

[9] L. Yang, R. Ege, Integrating Trust Management into Usage Control in P2P Multimedia Delivery, Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE08), p.411-416, Redwood City, CA, 2008.

[10] Open Handset Alliance, http://www.openhandsetalliance.com/. [accessed 10-5-10]

[11] Gutmann, P., 1999. The Design of a Cryptographic Security Architecture, *8th USENIX Security Symposium*, p. 153-168, Washington, D.C., 1999.

[12] Network Working Group, Diffie-Hellman Key Agreement Method, RFC: 2631, RTFM Inc., June 1999.

[13] The Legion of the Bouncy Castle Java Cryptography, http://www.bouncycastle.org/java.html. [acc. 10-5-10]

[14] R. Stewart (ed.), Stream Control Transmission Protocol, Request for Comments: 4960, IETF Network Working Group, Sept. 2007, http://tools.ietf.org/html/rfc4960. [accessed 10-5-10]