マルチメディア通信と分散処理 85−37
（1997. 11. 7）

キャッシュ可能なHTTPのアクセス制御手法

清水 亮博　山中 顕次郎
{akihiro, yamanaka}@slab.ntt.co.jp
NTT ソフトウェア研究所

概要

HTTPのトラフィック軽減やfirewall越しのWorld Wide Webアクセスのため、現在HTTPの代理キャッシュサーバが広くInternet上で用いられるようになっている。しかし、HTTPのアクセス制御はキャッシュできない上に、セキュリティ上の脅威である盗聴や偽造を防止することができない。本稿では暗号化を用いたHTTPのアクセス制御方式であるHTTP/SAE (HTTP with Simple Authentication Extension) を提案し、本方式がキャッシュ可能で盗聴や偽造を防止できることを示している。

A cacheable access control method of HTTP

Akihiro Shimizu,　Kenjiro Yamanaka
NTT Software Laboratories

Abstract

To reduce traffic of HTTP and to access World Wide Web over firewalls, proxy cache http servers becomes to be applied widely on the Internet. However, the access controlled traffic of HTTP cannot be cached, and cannot prevent violations of security - such as eavesdropping and forging. In this paper, we propose the HTTP/SAE (HTTP with Simple Authentication Extension), cacheable authentication method, which uses cryptography. And we show HTTP/SAE can be cached, safe from eavesdropping and forging.

1　はじめに

現在Internet上ではWorld Wide Web(以下WWW)による情報提供が盛んに行われるようになっている。このWWWは幅広い種類の情報を取り扱う能力を持ち、WWWサーバを用意するのみでInternet全体へ情報を発信可能であるなど優れた特徴を持ち、Internetの標準GUI (Graphic User Interface) の地位を築き上げた。

このWWWの普及に伴いHTTPのトラフィックが急激に増加しており、この負荷を軽減することが求められている。一般にWWWで扱う情報はユーザ毎に変化しないことから、代理キャッシュサーバによってトラフィックを軽減する方法がとられており、Harvest[2]やSquid[1]等広域分散型のキャッシュサーバも用いられはじめている。しかし、cgi等によりユーザ毎、セッション毎に内容が異なる場合や、アクセス制御を行っている場合など、キャッシュできない場合が存在する。

今後Internetの一般化が進むにつれて、有料情報サービスなどによるアクセス制御の利用が増えることが予想される。アクセス制御とは権限を持つユーザにはアクセスを許可し、権限を持たないユーザからのアクセスを遮断するための手法である。したがって不正なアクセスを遮断しなければならないが、HTTPのアクセス制御は現状では盗聴などの不正なアクセスを遮断できない。

そこで本稿では上で述べた問題点を解決した、HTTPのキャッシュ可能なアクセス制御手法を提案する。

2　HTTPのアクセス制御に必要な条件

アクセス制御とは、権限を持つユーザにはアクセスを許可し、権限を持たないユーザからのアクセスを遮断するための手法である。

HTTPのアクセス制御を考えたとき、不正アクセスの方法として考えられる手法は以下の通り。

盗聴 権限を持つユーザのトラフィックや、そのキャッシュを盗聴する。

偽造 (なりすまし) 偽造による不正アクセスには、以下の 2 つの方法がある。

ユーザ偽造 権限を持つ正規ユーザになりすまし、不正にアクセスを行う。

サーバ偽造 サーバになりすまし、権限を持つ正規ユーザからのアクセスを受け付けることで、正規ユーザになりすますための情報を 引き出す。

したがって、HTTP のアクセス制御は盗聴や偽造に対応できる必要がある。

またキャッシュ可能な HTTP のアクセス制御とは、上記の条件に加えて代理キャッシュサーバでキャッシュされた HTTP メッセージがそのまま他の正規ユーザに送ることができる、すなわちキャッシュ上のメッセージがが再利用可能であることである。

以上より、HTTP のアクセス制御方式が満たすべき条件は、

- 不正アクセスの遮断
- セキュリティ保護 (盗聴防止、偽造防止)
- 代理キャッシュサーバでキャッシュされた情報が再利用可能

である。

現在 HTTP で用いられている Basic Authentication[3][4] はユーザ名とパスワードが平文で通信されているため、盗聴されるとユーザ偽造が容易に実行できる。また HTTP メッセージも平文のため、盗聴が容易である。

Digest Authentication[5] は、Basic Authentication を改良したもので、ユーザ名、パスワードは暗号化されている。しかし HTTP メッセージが平文であるため、盗聴が容易であることは変わらない。

SSL(Secure Socket Layer)[1] は 1 対 1 通信で用いられる暗号化プロトコルで、盗聴や偽造を防止できる。しかしセッション毎に暗号鍵が異なるため、メッセージを代理キャッシュサーバにキャッシュしても再利用できず、トラフィックを低減することができない。

3 提案するアクセス制御手法

情報販売の一方式である Infoket 方式 [6][7][8] は、情報を対称鍵暗号方式で暗号化して自由に配布し、暗号鍵と引き替えに課金するものである (図 1)。

この方式の長所の一つとして、暗号化した情報はそのままでは利用できないので、キャッシュ可能であるというものがある。この長所を HTTP のアクセス制御に応用した方式を考案した。この方式を HTTP/SAE(Simple Authentication Extention) と呼ぶ。

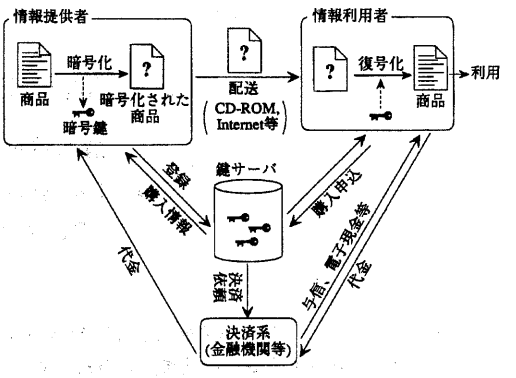

図 1: Infoket 方式

3.1 HTTP/SAE の概要

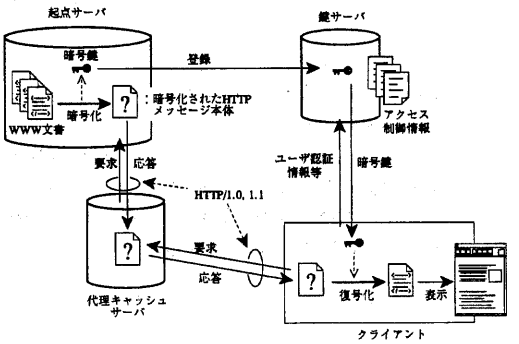

図 2: HTTP/SAE の概略

図 2に HTTP/SAE の概要を示す。

本方式は、メッセージ本体へのアクセスを制御する代りに、メッセージ本体を暗号化してその暗号鍵の授受でアクセスを制御する方式である。すなわち本方式では、資源に対するアクセス権を持つユーザとは、その資源を暗号化している暗号鍵を入手できるユーザとなる。

HTTP/SAE の動作の概略を以下に示す。

ユーザがクライアントを用いて資源を入手しようとすると、クライアントから通常の HTTP の要求が起点サーバ[2]に送られる。起点サーバでは対称鍵暗号方式で暗号

[1] SSL は Netscape Communications 社が開発したソケット層で暗号化するプロトコルであり、IETF にて SSL Ver.3 を元に TLS (Transport Layer Security) protocol として標準化作業中である。Internet draft は `draft-ietf-tls-protocol-03.txt`。

[2] RFC1945, RFC2068 等で定義されている origin server。資源を持つ、もしくは資源を生成する http サーバ。

化されたメッセージ本体をHTTPで送る。このときTransfer-Encodingヘッダには、暗号化の手法や鍵サーバのアドレスを表現する値、および暗号鍵のIDが入っている。

暗号化されたメッセージを受け取ったクライアントは、Transfer-Encodingヘッダより鍵サーバのIPアドレスを調べ、鍵IDとユーザ認証情報等を安全な方法で鍵サーバに送る。鍵サーバは受け取ったユーザ情報より、IDに該当する暗号鍵のアクセス権が存在しているか調べ、存在すれば安全な方法で暗号鍵を送り返す。クライアントは受け取った暗号鍵でメッセージを復号化し、ユーザに表示する。

3.2 アクセス制御の必要条件

本節ではHTTP/SAEが前述のHTTPのアクセス制御方式に必要な条件を満たしていることを示す。

不正アクセスを遮断 HTTP/SAEはアクセス制御の対象となるメッセージ本体が対称鍵暗号方式で暗号化されているので、暗号鍵を受け取れないとメッセージ本体を復号化して利用することができない。

メッセージを受け取る権限のあるユーザのクライアントは、Transfer-Encodingヘッダで表現される鍵サーバより鍵IDに該当する暗号鍵を入手できるので、メッセージを利用できる。

これに対し、権限のないユーザのクライアントには鍵サーバが暗号鍵を渡さないので、メッセージを利用できない。

以上より、本方式は権限のないユーザからのアクセスを排除できることが示せた。

セキュリティ HTTPメッセージは暗号化されているので盗聴は不可能である。HTTPサーバはクライアントとの通信でユーザ情報等セキュリティが必要な情報が流れないので、偽造防止は不要である。鍵サーバとユーザの偽造防止は、鍵を渡す際のプロトコルに電子署名等を用いて防止する。

キャッシュ可能 代理キャッシュサーバでキャッシュされたHTTPメッセージは暗号化されているので、権限のないユーザに利用されることはない。またキャッシュされたHTTPメッセージが別の正規ユーザに利用される場合、そのユーザは鍵サーバより同じ鍵を受け取って復号化して利用できる。以上より本方式は代理キャッシュサーバにてキャッシュされた情報が再利用可能であることが示せた。

以上より、本方式はHTTPのアクセス制御に必要な条件を満たしている。

3.3 鍵管理方式

HTTP/SAEでは、暗号化されたHTTPメッセージに応じた暗号鍵を鍵サーバと通信して入手し、メッセージを復号化する。この際のHTTPメッセージの暗号方式と、鍵サーバとクライアント間での鍵交換方式をあわせて鍵管理方式と呼ぶ。

鍵交換方式は、盗聴、偽造を防止できるものである必要がある。したがって、メッセージの暗号化やサーバ、クライアントの認証が必要である。HTTP/SAEではこれらの方式については定めず、鍵管理方式全体を選択可能とする。これは、強度や計算量などの理論的な条件だけでなく、既存のシステムとの親和性などの実際的な制約や、法律や特許などの社会的制約も考慮したとき、一つの方法に決定するのは無理が大きいと考えたためである。

鍵交換方式には、以下のような方法の組合わせが考えられる。

- ユーザ認証方式
  - ユーザ名とパスワードによる方法
  - 公開鍵暗号方式を用いた電子署名による方法
  - IC Card等ハードウェアを用いた方法, etc.
- 鍵交換時の秘密通信の方法
  - SSLを用いる方法
  - 通信を使わず、オフラインで行う方法
  - その他独自の方法 (Infoketプロトコル[6]など)

またHTTPメッセージの暗号アルゴリズムも様々なものが存在するので、条件にあったアルゴリズムを選択すべきである。

3.4 暗号化HTTPメッセージの表現方法

HTTPのメッセージが暗号化されていることを表現するために、本方式ではHTTP/1.1[4]で導入されたTransfer-Encodingヘッダを用いる。図3にTransfer-Encodingヘッダの拡張を拡張BNF記法で示す。

SAE-key-managementは、鍵管理方式の名前を示している。クライアントはこの名前に応じた鍵管理方式を用いてメッセージを復号化する。

SAE-key-augmentは鍵サーバのアドレスや、鍵の識別子、暗号の強度、情報販売の際には価格や電子現金の識別子など、SAE-key-managementで表現される鍵管理方式が必要とする引数を表現するのに用いられる。

例は ssl-auth という名前の鍵管理方式で、URL形式で鍵サーバ名: sslserv.ntt.co.jp と鍵のID: sae/key1.key が表現されている。

```
Transfer-Encoding     = "Transfer-Encoding" ":" transfer-coding-list

transfer-coding-list = 1#transfer-coding | SAE-auth-coding
transfer-coding       = "chunked" | transfer-extension
transfer-extension    = token （拡張用）

SAE-auth-coding     = "SAE" SAE-key-management #SAE-key-augment
SAE-key-management = token （鍵管理方式名）
SAE-key-augment     = token （鍵管理方式の引数）

例:
Transfer-Encoding: SAE ssl-auth https://sslserv.ntt.co.jp/sae/key1.key
```

注:
#x : xが任意個ならんだ空白で区切られたリスト
1#x : xが1つ以上任意個ならんだ空白で区切られたリスト

図 3: Transfer-Encoding ヘッダ

3.5 HTTP/SAE の特徴

HTTP/SAE の特徴を以下に挙げる。

キャッシュ可能なアクセス制御方式 2節で示した条件を満たす アクセス制御方式である。

鍵管理方式を必要に応じて選択が可能 用途に応じて、鍵管理方式を選択可能にしている。

既存の HTTP/1.1 の自然な拡張 HTTP を暗号化するために、新たなプロトコルを作成することも考えられる。しかし本方式では HTTP/1.1 で定義された Transfer-Encoding ヘッダを用いて暗号化を表現し、それ以外は HTTP/1.0, 1.1 をそのまま用いている。このため HTTP/SAE を実装する際には、新たなプロトコル解釈プログラムを作成する必要がなく、既存の HTTP/1.0, 1.1 の実装の変更で比較的容易に実装できる。

4 考察

4.1 代理キャッシュサーバにてアクセス制御を行う方法

HTTP/SAE 以外のキャッシュ可能なアクセス制御方式の実現方式を考える。

HTTP のアクセス制御は、起点サーバでアクセスの可否が決定されているため、代理キャッシュサーバにてキャッシュすることができない。したがって、代理キャッシュサーバでアクセス制御の判断を行い、SSL 等でセキュリティを保護する方法が考えられる (図 4)。

(1) ユーザがリクエストを送ると、(2) 代理キャッシュサーバはそのユーザのアクセス権の有無を起点サーバから入手した情報より調べ、該当するファイルの授受の可否を決定し、(3) 該当のファイル、又はエラーを返す。

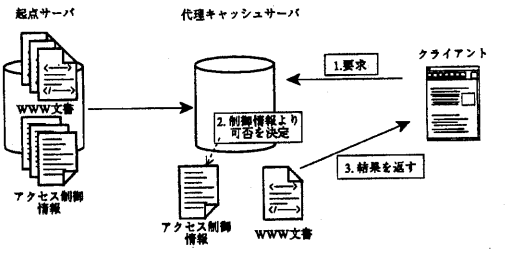

図 4: 代理キャッシュサーバによるアクセス制御

通信路上の盗聴、およびサーバ、クライアントの偽造は全ての通信を SSL 等で暗号化することで防止する。代理キャッシュサーバのキャッシュの盗聴は、代理キャッシュサーバの OS のアクセス制御で防止する。

この方式は 2節の条件を満たすものの、以下のような問題点がある。

アクセス制御情報によるトラフィック増大 HTTP/SAE ではアクセス制御情報によるトラフィック増大は、Transfer-Encoding ヘッダと鍵サーバとの通信に起因するものである。しかし、この両方とも数十 byte ～数百 byte 程度と比較的小さいものである。

これに対し、この方式ではアクセス制御情報を代理キャッシュサーバに送るが、アクセス制御情報はアクセス制御情報自体がトラフィックを増加させる恐れがある。

代理キャッシュサーバ 信用できない代理サーバが存在す

ると、盗聴や偽造を防止できないため、アクセス制御の意味がない。

したがって、全ての代理サーバの信頼性を調査し認証する必要があるが、これは Internet 全体では非現実的である。

以上の欠点のため、HTTP/SAE ではこの方式をとらなかった。

4.2 暗号化が本方式の性能に与える影響

HTTP/SAE は暗号化を用いているため、通常の HTTP アクセスより実行速度が低下する。そこで、実行速度の低下の程度を予測してみる。

性能を低下させる要因は、以下の 2 つである。

HTTP メッセージの復号化 HTTP/SAE では HTTP メッセージが暗号化して送られるため、起点サーバ、クライアントの両方に通常の HTTP アクセスより計算力が必要である。しかし、暗号化に用いられるのは対称鍵暗号方式のため、現在の計算機資源を考えると一般的には問題とならない。PDA 等の計算力の限られた計算機にクライアントが実装されている場合[3]は、暗号強度を落すことで対応可能である。

暗号鍵の授受 暗号鍵の授受の際、典型的なプロセスの概要を図 5 に示す。

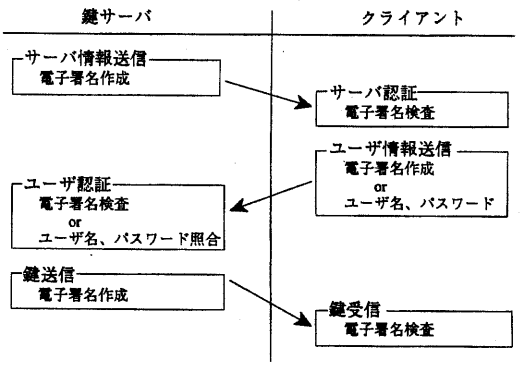

図 5: 暗号鍵の授受プロセスの概要

鍵サーバは自身の電子署名を作成し、クライアントに送信する。クライアントは電子署名を検査し、正しい鍵サーバであることを確認する。クライアントはユーザの認証のため、電子署名を作成、またはユーザ名とパスワードを得て暗号化し、鍵サーバへ送る。鍵サーバはそれらを検査してユーザを認証したのち、自身の電子署名を加えた暗号鍵を送る。クライアントは暗号鍵を受け取った後、電子署名を確認する。

以上のように、暗号鍵の授受には電子署名等計算力の必要な動作が存在する。また通信にも時間が必要である。これを防止するためには、暗号鍵の授受の回数を減らすことで対応できる。すなわち、サーバは同じ権限内の資源を同一の暗号鍵で暗号化し、クライアントでは受け取った暗号鍵を保存しておくことで、同一権限内の資源にアクセスする際には、鍵の授受が一回で済むようにできる。

以上の通り、HTTP/SAE の暗号化が性能に与える影響は一般的には問題にならないと予想される。

4.3 各要素の機能

HTTP/SAE の各要素に必要な機能を、以下に挙げる。

起点サーバ 起点サーバでは、メッセージを暗号化して送る必要がある。この時 cgi 等キャッシュ不能な資源はアクセス毎に暗号化する必要があるが、ファイルとして持っている資源はあらかじめ暗号化しておくことができる。なお、この場合ファイルの内容を変更する場合等には再暗号化が必要である。

鍵サーバ 鍵サーバの構成は鍵管理方式によって異なるが、共通に必要な機能は以下の通りである。

- ユーザ認証機構
- ユーザと暗号鍵の対照表
- 鍵の配送機構

また情報販売用の鍵サーバには、これ以外に課金機構などが必要である。

代理キャッシュサーバ HTTP/SAE は HTTP/1.1 をそのまま用いるので、基本的には変更は不要である。ただし、キャッシュ内では Transfer-Encoding を変更せず、そのまま保存する必要がある。

クライアント クライアントには鍵管理方式の機能、すなわち鍵サーバとの通信機能、およびメッセージの復号化機能が必要である。この鍵管理方式は用途などによって様々な方式が考えられるので、各鍵管理方式のモジュールをプラグイン方式にて追加可能にした上で、Transfer-Encoding の値に応じ

[3] リアルタイムで送られる動画像などでも影響が大きくなりうるが、本方式は通常の HTTP の範囲を対象にしているため、ここでは問題にしない。

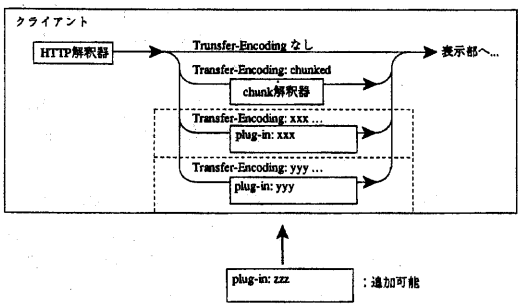

図 6: プラグイン方式による鍵管理方式の切り替え

たモジュールが呼び出されるようにすべきである(図6)。

クライアントは、HTTP メッセージの解釈が終了した後、Transfer-Encoding の第一引数の値によって該当するモジュールを呼び出す。各鍵管理モジュールは鍵サーバと通信を行い、暗号鍵を受け取って HTTP メッセージ本体を復号化し、クライアントに返す。クライアントは復号化されたメッセージ本体を受け取って表示する。

5 今後の課題

HTTP/SAE は現在実装を行っている。この実装を評価し、W3C 等へ提案することを予定している。

謝辞

本研究に取り組むきっかけを頂いた、NTT ソフトウェア研究所金井 敦 主幹研究員に感謝します。

参考文献

[1] Squid Internet Object Cache. http://squid.nlanr.net/Squid/.

[2] The Harvest Information Discovery and Access System. http://harvest.transac.com/.

[3] T. Berners-Lee, R. Fielding, and H. Nielsen. Hypertext Transfer Protocol – HTTP/1.0. Request for Comment 1945, May 1996.

[4] R. Fielding, J. Gettys, J. Mogul, H̃. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Request for Comment 2068, January 1997.

[5] J. Franks, P. Hallam-Baker, J. Hostetler, P̃. Leach, A. Luotonen, E. Sink, and L. Stewart. An Extension to HTTP : Digest Access Authentication. . Request for Comment 2069, January 1997.

[6] 森保健治, 明石修, 寺内敦, 三宅延久. 情報流通システムにおける鍵配送通信の構成法. マルチメディア通信と分散処理ワークショップ, October 1995.

[7] 明石修, 森保健治, 寺内敦. FleaMarket 方式による情報流通. マルチメディア通信と分散処理ワークショップ, October 1995.

[8] 金井敦, 三宅延久, 明石修, 生沼守英. マルチメディア情報流通システム (InfoKet). マルチメディア通信と分散処理研究会, May 1995.