DNS完全解説 の仕組み

第7回 TSIGおよびSIG(0)によるDNSメッセージの認証

馬場達也

前回は、クライアントからプライマリネームサーバ上のゾーンデータを動的に更新することが可能となる、DNSダイナミックアップデートについて解説した。今回は、DNSダイナミックアップデートによる更新を安全に行うために必要となる、TSIGおよびSIG(0)について解説する。

DNSダイナミックアップデートにおける認証の必要性

前回紹介したように、DNSダイナミックアップデートの機能を使用することによって、リモートのマシンからプライマリネームサーバ上のゾーンデータの更新を行うことが可能となる。しかし、単にプライマリネームサーバ上でDNSダイナミックアップデートの機能を有効にしただけでは、悪意を持った第三者がリソースレコードの内容を改ざんすることが可能となってしまう。このため、DNSダイナミックアップデートを許可する場合には、実際にゾーンデータを更新する前に、その要求元が正当なホストであることを確認し、そのメッセージの内容が通信路上で改ざんされていないことを確認する必要がある。

このために、DNSでは「DNSパケットの送信元の認証」と「通信路上でのDNSメッセージの改ざんの検出(メッセージ認証)」の2つの機能を実現するTSIG(Transaction Signature)およびSIG(0)というリソースレコードが定義されている。TSIGとSIG(0)の仕様は、それぞれ、RFC 2845およびRFC 2931に記述されている。セキュアDNSダイナミックアップデートについて記述しているRFC 3007では、DNSダイナミックアップデートメッセージにはTSIGまたはSIG(0)のどちらかを付加しなければならないと記述されている。

送信元の認証とメッセージ認証を実現するTSIG/SIG(0)

それでは、TSIGおよびSIG(0)での認証の仕組みについて説明しよう。TSIG/SIG(0)とも、「送信元の認証」と「メッセージ認証」を実現するものであるが、TSIGでは、「メッセージ認証コード(MAC:Message Authentication Code)」と呼ばれる認証技術を利用するのに対し、SIG(0)では、「デジタル署名」と呼ばれる認証技術を利用する。両者の大きな違いは、認証に使用する鍵(認証鍵)の管理と処理速度にある。メッセージ認証コードは、送信側と受信側が同じ鍵を使用する共通鍵ベースの認証技術であり、両者の間であらかじめ秘密に認証鍵を共有しておく必要がある。これに対してデジタル署名は、送信側と受信側が異なる鍵を使用する公開鍵ベースの認証技術であり、受信側が使用する鍵は第三者に公開してもかまわないため、メッセージ認証コードのように、あらかじめ秘密に鍵を共有しておくという手間が掛からない。これだけだと、デジタル署名のほうがすぐれているように思えるが、メッセージ認証コードには、デジタル署名と比較して格段に処理速度が速いという利点がある(表1)。

HMACによる認証でなりすましを防ぐ

TSIGでは、主にHMAC(Keyed-Hashing for Message Authentication Code)と呼ばれるメッセージ認証コードを使用する。HMACの仕組みは、RFC 2104に記述されており、TSIGだけでなく、IPsec(IP Security Protocol)やTLS(Transport Layer Security)、SSH(Secure Shell)などの多くのセキュリティプロトコルの認証アルゴリズムとしても利用されている。

HMACでは、認証対象となるメッセージに対し、秘密の認証鍵とハッシュ関数を使用して、MACと呼ばれる認証データを生成し、元のメッセージに付加する。ハッシュ関数とは、一方向性の関数であり、その

	認証方式	送信側の鍵	受信側の鍵	処理速度
TSIG	メッセージ認証コード	秘密	秘密	速い
SIG(0)	デジタル署名	秘密	公開	遅い

表1● TSIGとSIG(0)の違いをまとめた

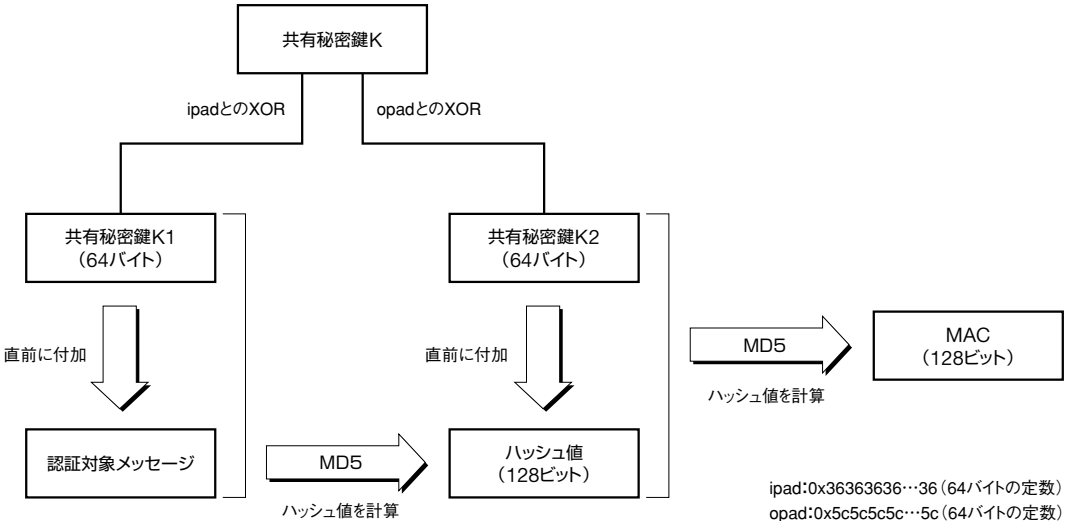

図1● HMAC-MD5によるMACの生成。共有秘密鍵を用いる

出力結果から元のメッセージを逆に計算することができない関数である。HMACでは、MD5やSHA-1などのさまざまなハッシュ関数を使用でき、使用するハッシュ関数によって、それぞれ「HMAC-MD5」「HMAC-SHA1」のように呼ばれる。

例として、HMAC-MD5によるMACの生成法を図1に示す。最初に、送信側と受信側であらかじめ秘密鍵Kを共有しておく。そして、秘密鍵Kとipadおよびopadとよばれる定数をそれぞれXOR演算することにより、K1、K2というサブ鍵を得る。送信側では、認証対象となるメッセージ(TSIGでは、DNSヘッダ以降の部分)に、サブ鍵K1を付加し、そのメッセージに対して、ハッシュ関数であるMD5を適用する。そして、その出力結果にサブ鍵K2を付加し、さらにMD5を適用する。そして、その結果をMACとし、メッセージとともに送信する。受信側では、受信したメッセージに対して送信側と同じ処理を行うことによってMACを生成し、受信側が生成したMACと送信側から送信されてきたMACを比較する。両者が一致すれば、「このメッセージは同じ認証鍵を持っている相手から送信され、通信路上でメッセージの内容が改ざんされていない」ことが確認できたことになる。

では、もし第三者が正当な送信者になりすましてメッセージを送信しようとするとどうなるのだろうか。その第三者は、MACを生成するために必要となる認証鍵を持っていないので、送信しようとするメッセージに対してMACを生成することができない。仮に任意の鍵でMACを生成したとしても、受信側で生成されるMACとは異なるため、認証に失敗する。また、第三者が通信路上でメッセージを改ざんしようとした場合も同様である。メッセージの内容を変更する場合には、MACの内容も合わせて修正しなければならない。しかし、メッセージを改ざんしようとする第三者は、MACを再生成するために必要な鍵を持っていないため、やはり、正しいMACを生成することができないのである。

デジタル署名による認証は鍵の交換が容易な利点を持つ

SIG(0)では、認証技術としてデジタル署名を使用する。メッセージ認証コードと比較すると、デジタル署名には鍵の管理が容易であるという利点があるが、処理速度が非常に遅いという欠点がある。

デジタル署名では、署名生成時と署名検証時に異なる鍵を使用する。署名生成用の鍵(秘密鍵)は、送信側で秘密に保持しておく必要があるが、もう一方の署名検証用の鍵(公開鍵)は、第三者に公開してもかまわない。このため、送信側から受信側に署名検証用の鍵を秘密に送信する必要がなく、鍵の管理が容易になる。

それでは、デジタル署名による認証の仕組みについて説明しよう。デジタル署名用のアルゴリズムとしては、DSA(Digital Signature Algorithm)やRSA/

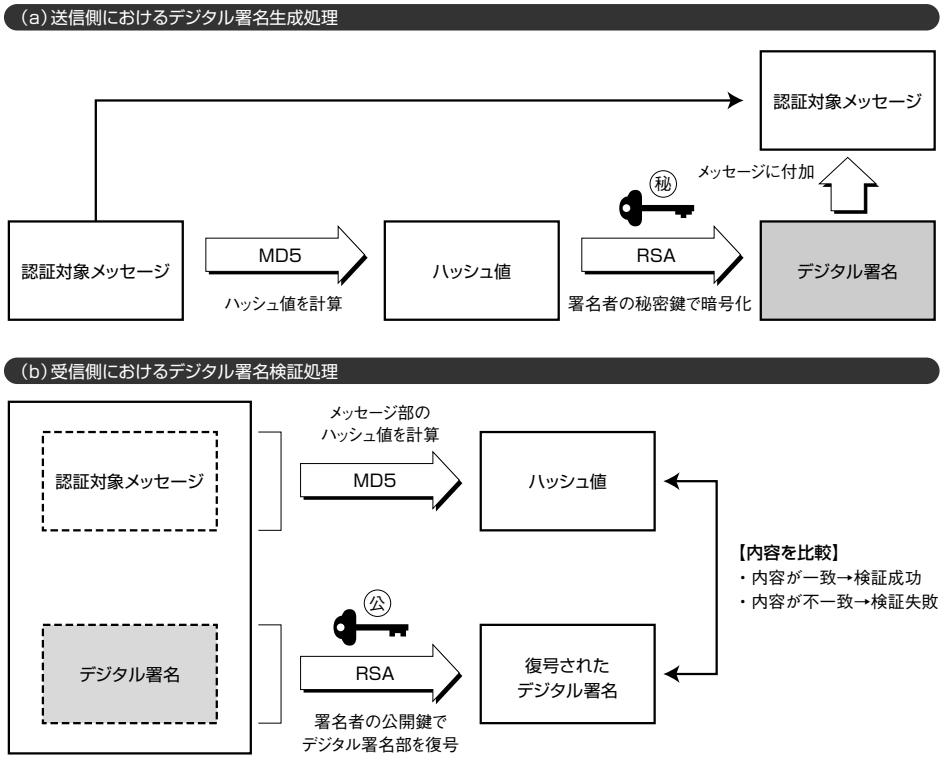

図2● デジタル署名による認証の仕組み（RSA/MD5の場合）。受信時に署名を復号して比較することで、メッセージの認証を行う

MD5などがある。図2に、デジタル署名アルゴリズムとして、RSA/MD5を使用した場合の例を示す。メッセージの送信側では、認証対象となるメッセージ(SIG(0)ではDNSヘッダ以降の部分）に対してハッシュ関数であるMD5を適用し、128ビットの長さのハッシュ値を生成する。このハッシュ値を、署名用の秘密鍵を使用してRSAで暗号化し、暗号化した結果を署名とする。そして、生成した署名をオリジナルのメッセージに付加して送信する。

受信側では、メッセージに付加されている署名を検証用の公開鍵を使ってRSAで復号し、その結果を、送信されてきたメッセージにMD5を適用した結果と比較する。ここで両者が一致した場合には、メッセージの内容が途中で変更されておらず、対応する署名用の秘密鍵を持っている者によってメッセージが作成されたことが確認できたことになる。もし、秘密鍵を持たない第三者が署名を行ったり、途中でメッセージを改ざんしたりした場合には、署名の検証に失敗する。

TSIG/SIG(0)を付加したDNSメッセージ

TSIGまたはSIG(0)を適用した場合は、DNSメッセージの最後の付加情報部分（Additional Section)に、それぞれTSIGリソースレコードまたはSIG(0)リソースレコードが付加される(図3)。TSIGリソースレコードおよびSIG(0)リソースレコードは、ゾーンデータファイルには記述されず、DNSパケット上にのみ存在する特殊なリソースレコードであり、このようなリソースレコードは「メタリソースレコード」と呼ばれる。

■TSIGリソースレコードのフォーマット

TSIGリソースレコードのフォーマットは図4のようになっている。

NAMEフィールドには、HMACで使用する鍵のIDが入り、TYPEフィールドには「250 (TSIG)」が入る。また、CLASSフィールドには「255 (ANY)」が入り、TTLフィールドは「0」となる。そして、RDLENGTHフィールドには、RDATAフィールド(Algorithm Nameフィールドから Other Dataフィールドまで）の長さがバイト単位で入る。

Algorithm Nameフィールドには、MACアルゴリズムのIDが入る。例えば、MACアルゴリズムがHMAC-MD5の場合は、「HMAC-MD5.SIG-ALG.REG. INT.」というドメイン名形式の文字列が入る(この文字列は、http://www.iana.org/assignments/tsig-algorithm-namesに記述されている)。Time Signedフィールド

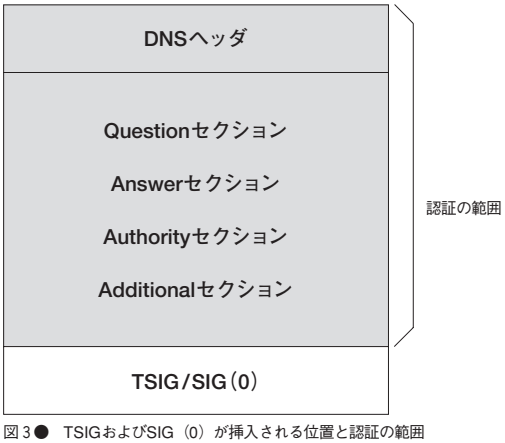

図3● TSIGおよびSIG（0）が挿入される位置と認証の範囲

にはMACを生成した時刻が入り、Fudgeフィールドには MACの有効期間（MAC生成時刻からのずれ）が秒単位で入る。MAC Sizeフィールドには MACの長さがバイト単位で入り、MACフィールドには MACデータが入る。そして、Original IDフィールドには DNSヘッダのメッセージIDが入り、Errorフィールドには、TSIG特有のエラーが発生した場合のエラーコードが入る。Other Lenフィールドには、Other Dataフィールドの長さがバイト単位で入るが、Other Dataフィールドは通常存在せず、この場合は「0」となる。

■SIG（0）リソースレコードのフォーマット

SIG（0）リソースレコードでは、DNSSEC（DNS Security Extensions）の仕様で定められているSIGリソースレコードと同じフォーマットを使用する。SIGリソースレコードのフォーマットは、RFC 2535に記述されており、図5のようになっている。

SIG（0）では、NAMEフィールドには、ドメイン名形式で「ルート」を表す「.」（1バイトの0）が入り、TYPEフィールドには「24（SIG）」が入る。また、CLASSフィールドには「255（ANY）」が入り、TTLフィールドには「0」が入る。RDLENGTHフィールドには、RDATAフィールド（Type Coveredフィールドから Signatureフィールドまで）の長さがバイト単位で入る。

Type Coveredフィールドには、SIG（0）では「0」が入る（ここが「0」となるため、SIG（0）と呼ばれる）。Algorithm Numberフィールドには、http://www.iana.org/assignments/dns-sec-alg に記述されているデジタル署名アルゴリズムの番号が入る。例えば、DSAを使用する場合には「3」が、RSA/MD5を使用する場合には「1」が入る。LabelsフィールドおよびOriginal TTLフィールドには、SIG（0）では「0」が入る。

Signature Expirationフィールドには署名の有効期限が入り、Signature Inceptionフィールドには署名時刻が入る。Key Tagフィールドには鍵のIDが入り、Signer's Nameフィールドには署名をしたホストのホスト名がFQDN（Fully Qualified Domain Name）で記述される。最後のSignatureフィールドには署名データが入る。

TSIGおよびSIG（0）では、署名の有効期間が設定されている。このため、送信側と受信側で時間を合わせておかないと、署名の有効期間の範囲外であると判断され、受信を拒否される可能性がある。この有効期間は、BINDでは署名時刻の±5分で設定されるため、クライアントとサーバとの間の時間のずれがその範囲内でなければならない。NTP（Network Time Protocol）などを利用して、クライアントとサーバとの間の時刻を合わせておくとよいだろう。

TSIGで使用する鍵をセットアップするTKEY

TSIGでは、MACの生成および検証で使用する認証鍵を、送信側と受信側であらかじめ秘密に共有しておかなければならない。これは、前もって認証鍵を送信側と受信側の設定ファイルに書き込んでおくことで実現できるが、長い間同じ鍵を使用していると、使用し

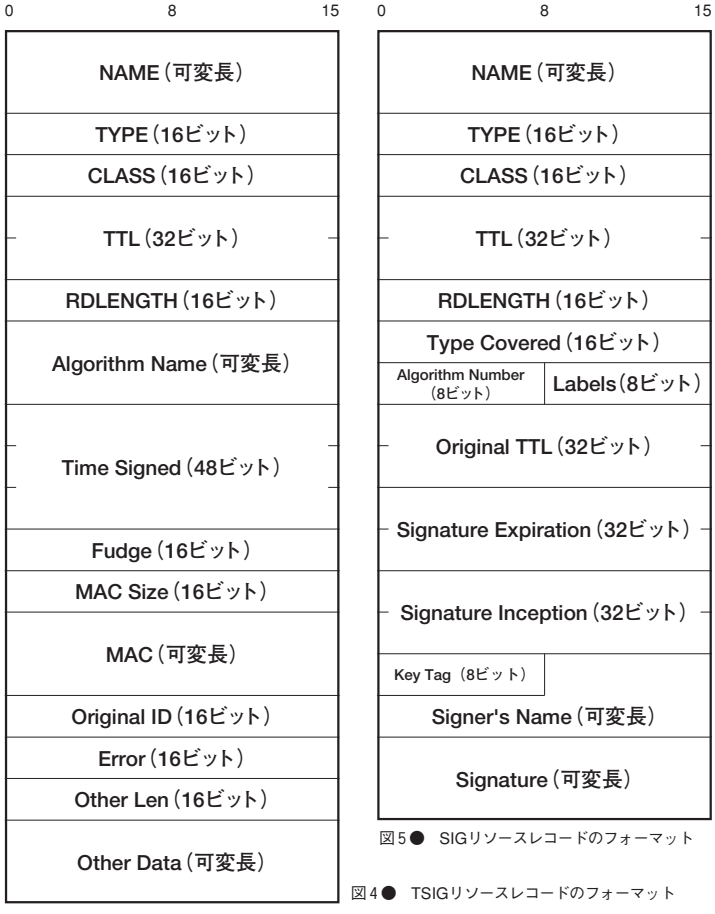

図5● SIGリソースレコードのフォーマット

図4● TSIGリソースレコードのフォーマット

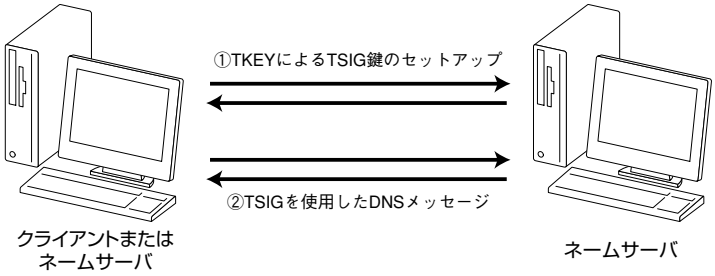

図6● TKEYによるTSIG鍵のセットアップ

```
//正引きゾーン
zone  "example.com" {
        type master;
        file  "example.com.zone";
        allow-update { key ddns.example.com.; };  ← この行を追加する
};
//逆引きゾーン
zone  "0.168.192.in-addr.arpa" {
        type master;
        file  "0.168.192.in-addr.arpa.zone";
        allow-update { key ddns.example.com.; };  ← この行を追加する
};
//TSIG鍵の設定
key ddns.example.com. {
        algorithm hmac-md5;
        secret "iSUuKtizDJEK/9ptgewTHQ==";        ← 生成した鍵を記述する
};
```

リスト1● TSIG付きDNSダイナミックアップデートの設定（named.confファイル）

ている鍵が第三者に解読されてしまう危険性がある。このため、TSIGで使用する鍵は定期的に変更したほうがよい。しかし、定期的に手動で鍵を変更するのは、非常に手間のかかる作業である。このため、共有している鍵を安全に変更するためのプロトコルとして、TKEY（Transaction Key）が定義されている。

TKEYの仕様は、RFC 2930に記述されている。TKEYを使用した場合は、図6のように、最初にTKEYによる認証鍵のセットアップが行われ、その認証鍵をTSIGが使用する。しかし、TKEYのメッセージをTSIGかSIG(0)を使用して認証しなければならない仕様になっており、初回には、手動でセットアップされた鍵を使用したTSIGもしくはSIG(0)を使用して、TKEYによる鍵のセットアップを行わなければならない。

BIND 8および9でのTSIGの設定

BIND 8および9では、TSIGが実装されている。また、SIG(0)については、メッセージの受信側の機能（署名の検証機能）のみが実装されており、TKEYについては、サーバ側の機能（TKEYリクエストを受信する側の機能）のみが実装されている。

ここでは、BINDで、DNSダイナミックアップデートの要求をTSIGで認証するための設定を紹介しよう。

最初に、TSIGで使用する認証鍵を生成する必要がある。この鍵は、「dnssec-keygen」コマンドで生成する。例えば、HMAC-MD5で使用する128ビットの鍵を生成するには次のように入力する（最後の「ddns.example.com.」というのは鍵IDであり、ほかの鍵のIDと重複しないように指定する）。

```
# /usr/sbin/dnssec-keygen -a HMAC-MD5 -b 128
-n HOST ddns.example.com.
```

すると、「Kddns.example.com.+157+07336.private」と「Kddns.example.com.+157+07336.key」という2つのファイルが作成される。TSIGで使用する鍵はどちらのファイルにも同じものが記述されている。例えば、「Kddns.example.com.+157+07336.private」ファイルには次のような形式でTSIG用の鍵が記述されている。

```
$ cat Kddns.example.com.+157+07336.private
Private-key-format: v1.2
Algorithm: 157（HMAC_MD5）
Key: iSUuKtizDJEK/9ptgewTHQ==
                ↑
        「iSUuKtizDJEK/9ptgewTHQ==」が鍵
```

そして、TSIGが付加されたDNSダイナミックアップ

```
// 正引きゾーン
zone  "example.com" {
        type master;
        file  "example.com.zone";
        allow-transfer { key axfr.example.com.; }; ← この行を追加する
};
// 逆引きゾーン
zone  "0.168.192.in-addr.arpa" {
        type master;
        file  "0.168.192.in-addr.arpa.zone";
        allow-transfer { key axfr.example.com.; }; ← この行を追加する
};
// TSIG鍵の設定
key axfr.example.com. {
        algorithm hmac-md5;
        secret "nAgPY1cTo8HJ/FXw1waaow==";          ← 生成した鍵を記述する
};
```

リスト2● プライマリネームサーバにおけるゾーン転送の設定（named.confファイル）

```
// TSIGを使用するサーバの設定
server 192.168.0.10 {        ← プライマリネームサーバのIPアドレスを記述する
        keys { axfr.example.com.; };        ← TSIG鍵を指定する
};
// TSIG鍵の設定
key axfr.example.com. {
        algorithm hmac-md5;
        secret "nAgPY1cTo8HJ/FXw1waaow==";  ← 生成した鍵を記述する
};
```

リスト3● セカンダリネームサーバにおけるゾーン転送の設定（named.confファイル）

デートをプライマリネームサーバで許可するために、BINDの設定ファイルである「named.conf」ファイルのzoneステートメントにおいて、リスト1のように設定する。

DNSダイナミックアップデートのクライアントプログラムとしてnsupdateを使用する場合は、次のように「-k」オプションでTSIG鍵を含んだ鍵ファイルを指定することで、TSIG付きのDNSダイナミックアップデートメッセージを発行するようになる。

```
# nsupdate -k Kddns.example.com.+157+07336
.private
```

また、TSIGでゾーン転送の要求元を認証したい場合には、プライマリサーバおよびセカンダリネームサーバにおいてリスト2およびリスト3のように設定する。

次回は、DNSのIPv6への対応について説明する。

NTTデータ 馬場達也

●今回の内容に関連するRFC

RFC 2104 “HMAC: Keyed-Hashing for Message Authentication”
RFC 2535 “Domain Name System Security Extensions”
RFC 2845 “Secret Key Transaction Authentication for DNS (TSIG)”
RFC 2930 “Secret Key Establishment for DNS (TKEY RR)”
RFC 2931 “DNS Request and Transaction Signatures(SIG(0)s)”
RFC 3007 “Secure Domain Name System (DNS) Dynamic Update”