DNS完全解説
の仕組み
第9回 DNSSECでゾーンデータの改ざんを検知する

馬場達也

前回は、IPv6ネットワークを構築する際に必要となる、DNSのIPv6対応について解説した。今回は、ネームサーバが保持するゾーンデータに署名を施すことにより、ゾーンデータの改ざんを検知するDNSSEC（DNS Security Extensions）の仕組みについて解説する。

ゾーンデータの改ざんを検知するDNSSEC

DNSSEC（DNS Security Extensions）は、権威ネームサーバが保持するゾーンデータ内のリソースレコードに署名を施し、リソースレコードを取得したクライアントやローカルネームサーバがその署名を検証することによってリソースレコードの内容が改ざんされていないことを確認する技術である。

本連載の第7回で紹介したTSIG（Transaction Signature）およびSIG(0)を利用すれば、ネットワーク上を流れるDNSメッセージの改ざんやネームサーバのなりすましを検知することができる。しかし、悪意のある者がネームサーバに侵入してゾーンデータ自身を書き換えてしまった場合には、TSIGやSIG(0)を利用しても、これを検知することはできない。DNSSECを利用すると、このようなゾーンデータ自身の改ざんを検知できる。

ただし、DNSSECが完全に機能するためには、アクセスするすべてのネームサーバがDNSSECに対応している必要がある。現在は、まだルートネームサーバなどの上位のネームサーバがDNSSECに対応していないため、実際にはDNSSECを完全な形で利用することができない。しかし、DNSSECを実際に利用するための議論や実験が現在活発に行われており、近い将来にルートネームサーバなどの上位のネームサーバにDNSSECが導入されていくだろう。将来のためにも、ここでDNSSECの仕組みについてきちんと理解しておこう。

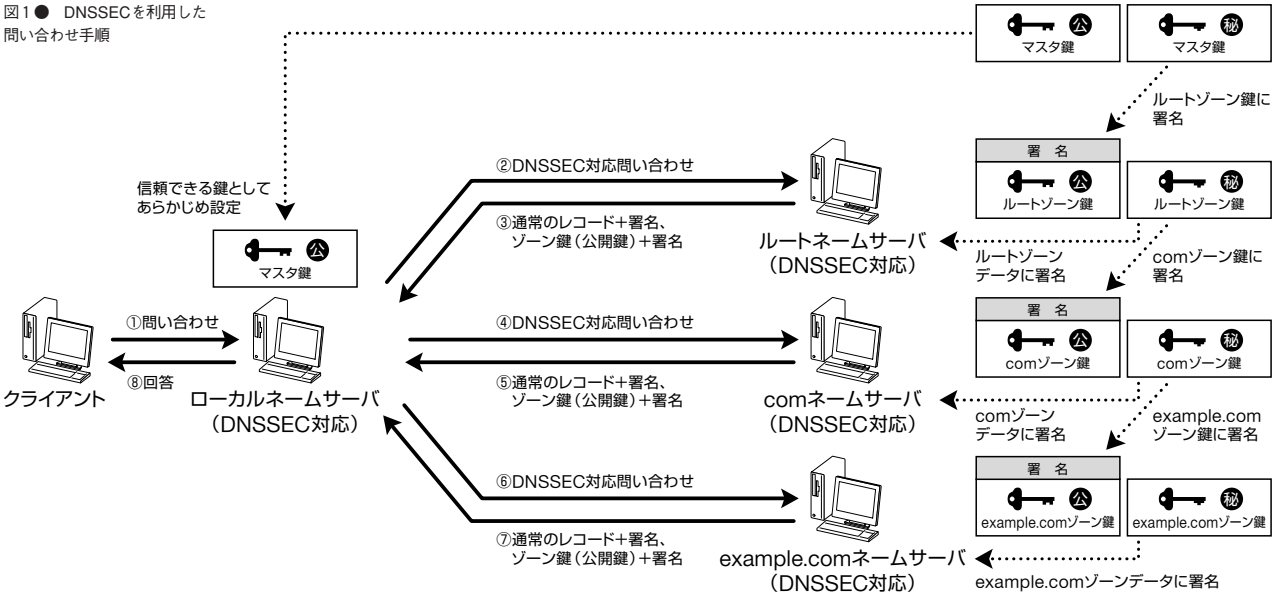
DNSSECの動作を理解する

DNSSECの仕様は、RFC 2535、3008、3090、3445に記述されている。DNSSECでは、ゾーンごとに公開鍵と秘密鍵のペア（これを「ゾーン鍵」という）を保持し、その秘密鍵を使用して、ゾーンデータ中の各リソースレコードに対して署名を施す。問い合わせに対する返答としてリソースレコードを受け取った側は、リソースレコードの署名をそのゾーンの公開鍵を使用して検証することによって、リソースレコードの内容が改ざんされていないことを確認できる。また、各ゾーンの公開鍵は、上位ゾーンの秘密鍵によって署名が施されているので、検証に使用する公開鍵自身が正しいかどうかは、その公開鍵の署名を上位ゾーンの公開鍵で検証することにより確認できる。ただし、ルートゾーンの公開鍵の署名を検証するための公開鍵（マスタ鍵）だけは、信頼できる鍵としてあらかじめ検証する側が保持しておく必要がある。

それでは、クライアントからexample.comゾーンのリソースレコードを、DNSSECに対応したローカルネームサーバを経由して問い合わせる場合の動作を図1を用いて説明しよう。実際には、ルートネームサーバやcomネームサーバ（gTLDネームサーバ）はDNSSECに対応していないが、ここでは、DNSSECに対応していると仮定して説明を行う。

DNSSECに対応したローカルネームサーバがクライアントから問い合わせを受けると（①）、ローカルネームサーバは、自身がDNSSECに対応していることを示した問い合わせをルートネームサーバに対して発行する（②）。自身がDNSSECに対応していることを示すためには、問い合わせのDNSメッセージの付加情報部（Additional Section）に、「DNSSEC OK（DO）」フラグをONにした「OPTリソースレコード」を付加する（このDOフラグはRFC 3225で規定されている）。DNSSEC対応問い合わせを受信したルートネームサーバは、回答として返答するリソースレコードとともに、

図1● DNSSECを利用した問い合わせ手順



各リソースレコードの署名と署名を検証するためのルートゾーンの公開鍵、ルートゾーンの公開鍵の署名を返す(③)。ローカルネームサーバには、あらかじめ信頼できる鍵として、ルートゾーンの公開鍵の署名を検証するためのマスタ鍵が設定されているので、そのマスタ鍵を使用してルートゾーンの公開鍵の署名を検証する。検証に成功すれば、そのルートゾーンの公開鍵を使用して各リソースレコードの署名を検証する。すべての署名の検証に成功すれば、次にローカルネームサーバはcomネームサーバに対してDNSSEC対応問い合わせを行う(④)。すると、comネームサーバはルートネームサーバと同様に、回答として返答するリソースレコードとともに、各リソースレコードの署名と、署名を検証するためのcomゾーンの公開鍵、そしてcomゾーンの公開鍵の署名を返す(⑤)。ローカルネームサーバは、すでに持っているルートゾーンの公開鍵を使用してcomゾーンの公開鍵の署名を検証し、検証に成功すれば、そのcomゾーンの公開鍵を使用して各リソースレコードの署名を検証する。すべての署名の検証に成功すれば、ローカルネームサーバは次にexample.comネームサーバに対してDNSSEC対応問い合わせを行う(⑥)。するとexample.comネームサーバは、同様に、回答として返答するリソースレコードとともに、各リソースレコードの署名と、署名を検証するためのexample.comゾーンの公開鍵、example.comゾーンの公開鍵の署名を返す(⑦)。ローカルネームサーバは、さきほど取得したcomゾーンの公開鍵を使用してexample.comゾーンの公開鍵の署名を検証し、検証に成功すれば、そのexample.comゾーンの公開鍵を使用して各リソースレコードの署名を検証する。すべての署名の検証に成功すれば、正しい回答が得られたことになるのでその結果をクライアントに返す(⑧)。

タイプ	意味
KEY	公開鍵を格納するためのリソースレコード
SIG	署名を格納するためのリソースレコード
NXT	次ドメイン名を示すためのリソースレコード

表1● DNSSECで使用するリソースレコードのタイプ

DNSSECで使用するリソースレコード

DNSSECでは、公開鍵を格納するためのKEYリソースレコード、リソースレコードの署名を格納するためのSIGリソースレコード、そして、リソースレコードが存在しないことを示すためのNXTリソースレコードが新たに定義されている(表1)。

●KEYリソースレコード

KEYリソースレコードは、公開鍵を格納するためのリソースレコードである。KEYリソースレコードの書式は次のようになっている。

＜owner＞ ＜ttl＞ ＜class＞ KEY ＜flags＞ ＜protocol＞ ＜algorithm＞ ＜public key＞

＜owner＞にはホスト名（正規名）を記述し、＜ttl＞には、このリソースレコードのキャッシュの有効期間を秒単位で記述する。＜ttl＞が省略された場合には、直前の$TTLでセットされたデフォルトの有効期

間がセットされる。<class>には、ネットワーククラスを記述する。インターネットでは「IN」と記述し、実際にはこれ以外には使用されていない。<class>が省略された場合には、自動的に「IN」がセットされる。

<flags>には、鍵の種類を指定するための16ビットの長さのフラグを10進数に直したものが入る。RFC 2535ではいくつかのフラグが定義されていたが、RFC 3445で、左から8ビット目の「ゾーン鍵フラグ」しか使用しないように変更されている。「ゾーン鍵フラグ」が「1」の場合は記述されている鍵がゾーン鍵であることを示し、「0」の場合は、例えばSIG(0)で使用するホスト鍵などのゾーン鍵以外の鍵であることを示す。したがって、<flags>には、ゾーン鍵の場合は「256」が入り、ゾーン鍵以外の場合は「0」が入る。

<protocol>には、この鍵を使用するプロトコルの番号が入る。RFC 2535では、KEYリソースレコードの鍵はDNSSECのほかに、TLS、S/MIME、IPsecなどでも利用できるように設計されていたが、RFC 3445で仕様が改定され、KEYリソースレコードはDNSSECでのみ利用することになった。このため、このフィールドには必ずDNSSECのプロトコル番号である「3」が入る。

<algorithm>には、http://www.iana.org/assignments/dns-sec-algに記述されている、公開鍵が使用する署名アルゴリズムの番号が入る(表2)。<public key>には、base64でエンコーディングした公開鍵が入る。

例えば、署名アルゴリズムとしてDSA/SHA-1を使用し、鍵の長さが1,024ビットのexample.comゾーンのゾーン鍵を格納したKEYリソースレコードはリスト1のようになる。

表2● DNSSECで使用される署名アルゴリズム

アルゴリズム番号	署名アルゴリズム	仕様
1	RSA/MD5	RFC 2537
2	Diffie-Hellman	RFC 2539
3	DSA/SHA-1	RFC 2536
5	RSA/SHA-1	RFC 3110

リスト1● 署名アルゴリズムとしてDSA/SHA-1を使用し、鍵の長さが1,024ビットのexample.comゾーンのゾーン鍵を格納したKEYリソースレコード

```
example.com. 172800  KEY  256 3 3 (
                          CPQ4z1GPtlwH/X90MO0HEsRYR/gfgwicOjK2
                          niyWFilsivyMlZSV3iZ/SFAHhXPG4BYcCi71
                          eZg7X6JLEVv5rKBjVS5IbacXCO5qJUY6RkoJ
                          VLJVmZZYef12PCsB/fMXQ1CADR/uTTmPqeuj
                          8m0lK59c4L3nD16okiyF1JHi1Tsk7t1BPtsZ
                          61h6Ks37p+aueg/iJ+1325Rryr6qC/7iUlXv
                          LaYAxy7qa1TRwOvpYoXxotQ37Xs4eUIQ3KfC
                          YQwPaW0fWe0EvtanpkkDlKUwuWPoNwrLvmUF
                          3OAwyleLEQiCxepPYmp3EWTlUh6tIVo9ygj4
                          GF3yk7+IOFbmPGrjmgQmC0TSpMwdukEzZBlf
                          aCI/u3MUdSgX+uslYw/3tpORlrMP9muhDiUl
                          SWe5MURREXlS9/OFTFqETW6nExIK5q4b9ls9
                          BzTeGjPI3mFoHNltCHwXL8qA4VgMSkTHIVkO
                          3K90oagS/mYxfFng4DuOlITqEPd5LftZKLJB
                          QzD/7ykIVPFMQmfmHxEjKii+vkcxQN0SF9kB
                          )
```

●SIGリソースレコード

SIG(Signature)リソースレコードは、署名を格納するためのリソースレコードである。SIGリソースレコードの書式は次のようになっている。

<owner> <ttl> <class> SIG <type covered> < algorithm> < labels> < original TTL> < signature expiration> <signature inception> < key tag> <signer's name> <signature>

<type covered>には、AやMXなどの署名対象となるリソースレコードのタイプが入る。また、<algorithm>には署名アルゴリズムの番号が入り、<labels>には<owner>で記述したドメイン名のラベルの個数が入る(例えば<owner>が「www.example.com.」であれば、<labels>には3が入る)。そして、<original TTL>には、署名対象のリソースレコードのTTL値が入り、<signature expira tion>には署名の有効期限が、<signature inception>には署名時刻が入る。署名の有効期限や署名時刻は「YYYYMMDDhhmmss」の形式でUTC(協定世界時)で記述される(YY YY=年、MM=月、DD=日、hh=時、mm=分、ss=秒)。また、<key tag>には、署名を検証するゾーン鍵の鍵タグが入る。この鍵タグは、公開鍵の内容から計算され、同じゾーンに複数のゾーン鍵が存在する場合に、それらを識別するために使用される。そして、<signer's name>には、署名を行ったゾーン鍵のゾーン名が入り、<signature>には、base64でエンコーディングした署名が入る。

例えば、「www.example.com」のAレコードに対してDSA/SHA-1で署名をした結果を格納したSIGリソースレコードはリスト2のようになる。

●NXTリソースレコード

NXT(Next Domain)リソースレコードは、あるリソースレコードが存在しないことを示す

```
www.example.com.  86400  IN  A    192.168.0.30
                  86400  IN  SIG  A 3 3 86400 20030204094924 (
                                  20030105094924 49073 example.com.
                                  CEEskTxmzq2fdoe2jnG+OUcrLRhoUONPC2no
                                  rYipYJbtO3OIGWZblrY= )
```

リスト2● www.example.comのAレコードに対してDSA/SHA-1で署名をした結果を格納したSIGリソースレコード

```
a.example.com.  86400  IN  A    192.168.0.50
                86400  IN  SIG  A 3 3 ...
                86400  IN  NXT  c.example.com. A SIG NXT
                86400  IN  SIG  NXT 3 3 ...
```

リスト3● NXTリソースレコードの記述例

ためのリソースレコードである。NXTリソースレコードでは、アルファベット順で自身のドメイン名の次に存在するドメイン名を指定することによって、その間のドメイン名に対するリソースレコードが存在しないことを示す。このNXTリソースレコードの内容が正しいことは、NXTリソースレコードに対するSIGリソースレコードを検証することによって確認できる。NXTリソースレコードの書式は次のとおりである。

＜owner＞ ＜ttl＞ ＜class＞ NXT ＜next domain name＞ ＜type＞ ＜type＞...

＜owner＞には、ゾーンデータ中に存在するドメイン名が入り、＜next domain name＞には、アルファベット順で次に存在するドメイン名が入る。＜type＞には、＜owner＞で記述したドメイン名に対して存在するリソースレコードのタイプを列挙する。

例えば、「a.example.com」というドメイン名に対してすでにAおよびSIGリソースレコードが存在し、アルファベット順で「a.example.com」の次に存在するドメイン名が「c.example.com」であった場合には、NXTリソースレコードはリスト3のようになる。これにより、アルファベット順で「a.example.com」と「c.example.com」の間に位置する「b.example.com」というドメイン名に関するリソースレコードが存在しないことを示すことができる。また、同時に、「a.example.com」というドメイン名に対しては、A、SIG、NXTの各リソースレコード以外には存在しないことも示される。

DNSSEC実装済みのBIND 9で権威ネームサーバを設定する

BIND 9ではすでにDNSSECの機能が実装されている。そこで、ここでは、BIND 9による権威ネームサーバをDNSSECに対応させるための方法を紹介する。

①ゾーン鍵の生成

はじめに、プライマリネームサーバ上でゾーンの鍵ペアを生成する。ゾーンの鍵ペアは、「dnssec-keygen」コマンドで生成することができる。例えば、example.comゾーンで使用する1,024ビットのDSA用の鍵ペアを生成する場合には次のようにする。

```
# /usr/sbin/dnssec-keygen -a DSA -b 1024 -n ZONE example.com.
Kexample.com.+003+49073
```

入力するコマンド

生成された鍵ペアのファイル名

②上位ゾーンへの送付用鍵ファイルの作成

次に、生成した公開鍵を上位ゾーンの鍵で署名してもらうために、「dnssec-makekeyset」コマンドを使用して、生成した公開鍵に対して自身の秘密鍵で署名した鍵ファイルを作成する。これにより、「keyset-[ゾーン名]」というファイルに、ゾーンの公開鍵を含んだKEYリソースレコードと、自身の秘密鍵でKEYリソースレコードを署名したSIGリソースレコードが格納される。

```
# /usr/sbin/dnssec-makekeyset -t 172800 Kexample.com.+003+49073.key
keyset-example.com.
```

入力するコマンド

鍵のTTL

署名対象公開鍵ファイル

生成されたファイル

③上位ゾーンの鍵によるKEYリソースレコードへの署名

作成した「keyset-[ゾーン名]」ファイルを上位ゾーン（この例ではcomゾーン）の管理者に送り、そこで、上位ゾーンの秘密鍵を使用してKEYリソースレコードに署名をしてもらう。KEYリソースレコードへの署名は、「dnssec-signkey」コマンドで行うことができる。

署名を行った結果は、「signedkey-[ゾーン名]」ファイルに格納される。

```
# /usr/sbin/dnssec-signkey keyset-example.com. Kcom.+003+61511.private
                          ↑署名対象鍵ファイル  ↑署名用秘密鍵ファイル
signedkey-example.com.
↑生成されたファイル
```
入力するコマンド

④署名された鍵のゾーンファイルへの取り込み

上位ゾーンの管理者から送られてきた「signedkey-[ゾーン名]」には、example.comゾーンのゾーン鍵が格納されたKEYリソースレコードと、そのKEYリソースレコードに対して上位ゾーンの秘密鍵で署名をしたSIGリソースレコードが含まれているので、このファイルの内容をゾーンファイルに取り込む。

```
# cat signedkey-example.com. >> example.com.zone
```

⑤ゾーンファイルへの署名

「dnssec-signzone」コマンドを使用して、ゾーンファイルに対して署名を施す。これにより、各リソースレコードに対してSIGリソースレコードとNXTリソースレコードが付加される。署名済みのゾーンデータファイルは、「.signed」という拡張子が付いた名称で作成される。

```
# /usr/sbin/dnssec-signzone -o example.com. example.com.zone
                            ↑ゾーン名  ↑ゾーンファイル
example.com.zone.signed
↑署名済みゾーンファイル
```

BIND 9ローカルネームサーバでマスタ鍵を設定する

BIND 9が動作しているローカルネームサーバでDNSSECの検証を行えるようにするには、ルートゾーンのマスタ鍵（公開鍵）を/etc/nemed.confファイルの「trusted-keys」ステートメントで記述する（ただし、現在はルートゾーンのマスタ鍵は存在しないので、実際にはDNSSECに対応している最上位のゾーンの公開鍵を設定する）。ここでは、リスト4のように「ゾーン名」「フラグ（ゾーン鍵の場合は256となる）」「プロトコル（必ず3となる）」「アルゴリズム番号」「公開鍵（base64でエンコーディングしたもの）」の順で記述する。

digコマンドで確認する

DNSSECの機能を利用するためには、クライアント側がDNSSECに対応していることをサーバ側に伝えなければならない。これは、クライアントからのクエリーにOPTリソースレコードを付加し、「DNSSEC OK（DO）」フラグをONにすることで行われる。digでは、オプションで「+dnssec」と指定することにより、このフラグをONにしたクエリーを発行できる。

最初に、存在するリソースレコードを問い合わせた場合のdigの結果をリスト5に示す。回答部（ANSWER SECTION）には、通常の回答とともに、そのリソースレコードに対するSIGリソースレコードが付加される。この署名は、付加情報部（ADDITIONAL SECTION）に含まれるKEYリソースレコード中の公開鍵を使用して検証することができる。また、example.comゾーンの公開鍵は、その上位のcomゾーンの公開鍵によって署名されていることがわかる。

次に、存在しないリソースレコードを問い合わせた場合のdigの結果をリスト6に示す。この場合は、権威部（AUTHORITY SECTION）に、SOAリソースレコードのほかにNXTリソースレコードとこれらのリソースレコードに対するSIGリソースレコードが含まれる。このNXTリソースレコードでは、「ns2.example.com」の次に存在するドメイン名は「www.exmaple.com」であり、アルファベット順でその間に位置する「sykes.example.com」が存在しないことを示す。

鍵の署名手続きを簡略化するDSリソースレコード

RFC 2535に記述されているDNSSECでは、鍵を上位ゾーンの管理者に送って署名をしてもらい、それを送り返してもらうという手間がかかる。そこで、現在は、この手間を軽減させるための新しいDNSSECの仕様が議論されている。新しい仕様では、各ゾーンでは、ゾーン鍵のほかにゾーン鍵に署名を行うための鍵署名鍵を保持する（図2）。これにより、ゾーンの管理者は、ゾーン鍵を変更した場合でも、その鍵に自身で保持する鍵署名鍵で署名をすればよく、上位ゾーンに署名をしてもらう必要がなくなる。この鍵署名鍵は、上位ゾーンのDS（Delegation Signer）リソースレコードによって承認してもらうことにより、その鍵の正当性が保証される。このDSリソースレコードのフォーマットは次のようになっている。

リスト4● BIND 9の/etc/named.confファイルの設定

```
trusted-keys {
        "." 256 3 3 "CJl/EAFaOS2j8pFSHeQFaIC8aT2jqjfs0TY3
                     7wRRzjyhDTqjtm1YFAod CXJMwCstxLKRHyE
                     sJtX4G7P3ocKAx3xzXPSGOsE118HLXyHf64h
                     0HP/+ KIX/6jKnzhtIWSjiTEuK4hzESpNdkp
                     IpxaDCH0VxSvzxpw7ZWkBoYGYt 7mnh2Oyfc
                     JLwLAomuj1j0FxKC0sOeVOf5nmcPy5ZuNj63
                     QgSSJnjZ7Zp 0m5TVIMiZr9cncD6QDpyXNJ1
                     UyhWzpG0a/DfmIpaZd9E1Sa1d+oyZrpc JlG
                     iL+Em6Yt+LxSdFVcN39gjk8bnsTEEB/KiSwM
                     IqGBl///oEdiSb93w I5PmM1W6gk/iFnmE3N
                     sRiTZRKFHYsrbW9kfRnj+Dmj+3PKYxXeCl0B
                     jg T3Y/KGCfnGjnT4vn1jx7J1tmtRwMYveAU
                     o5edr/dti9OWURY/7SW+S7S q3P8oFSWyntX
                     vXoXmwKEG0zO6YfivZ1MqfNDyC0z/pUqpy+J
                     sLErXco7 ew1pltk9m5L+TeRpJw7olAvgfzn
                     LiUzI0NE4";
};
```

リスト5● digコマンドによる問い合わせの結果（リソースレコードが存在した場合）

```
$ dig @ns1.example.com. +dnssec +norec www.example.com.

; <<>> DiG 9.2.1 <<>> +dnssec +norec www.example.com.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4374
;; flags: qr aa; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.example.com.            IN      A

;; ANSWER SECTION:
www.example.com.      86400   IN      A       192.168.0.30
www.example.com.      86400   IN      SIG     A 3 3 86400 20030204094924 20030105094924 49073 examp
le.com. CEEskTxmzq2fdoe2jnG+OUcrLRhoUONPC2norYipYJbtO3OIGWZblrY=

;; AUTHORITY SECTION:
example.com.          86400   IN      NS      ns1.example.com.
example.com.          86400   IN      NS      ns2.example.com.
example.com.          86400   IN      SIG     NS 3 2 86400 20030204094924 20030105094924 49073 exam
ple.com. CGQ5UXXYEvD0kQa0HeH4G+pzq2QPDqTCJu3MTnUnmq4DmwsEMcu7jSc=

;; ADDITIONAL SECTION:
ns1.example.com.      86400   IN      A       192.168.0.10
ns2.example.com.      86400   IN      A       192.168.0.11
ns1.example.com.      86400   IN      SIG     A 3 3 86400 20030204094924 20030105094924 49073 examp
le.com. CD9FcVXBlyKbrPhtGnbmoUQPMdQgllJFSgIPt0rN1OfwbSelhLDvTZA=
ns2.example.com.      86400   IN      SIG     A 3 3 86400 20030204094924 20030105094924 49073 examp
```

リスト5続き

```
le.com. CHoYwW2bJItp9dnsnq+C2nxJfXfUIVaHrYnmE8Z+3d8tBm1LTrm+n3Q=
example.com.          172800  IN      KEY     256 3 3 CPQ4z1GPtlwH/X90MO0HEsRYR/gfgwicOjK2niyWFilsi
vyMlZSV3iZ/ SFAHhXPG4BYcCi71eZg7X6JLEVv5rKBjVS5IbacXCO5qJUY6RkoJVLJV mZZYef12PCsB/fMXQ1CADR/uTTmP
qeuj8m0lK59c4L3nD16okiyF1JHi 1Tsk7t1BPtsZ61h6Ks37p+aueg/iJ+1325Rryr6qC/7iUlXvLaYAxy7q a1TRwOvpYoX
xotQ37Xs4eUIQ3KfCYQwPaW0fWe0EvtanpkkDlKUwuWPo NwrLvmUF3OAwyleLEQiCxepPYmp3EWTlUh6tIVo9ygj4GF3yk7+
IOFbm PGrjmgQmC0TSpMwdukEzZBlfaCI/u3MUdSgX+uslYw/3tpORlrMP9muh DiUlSWe5MURREXlS9/OFTFqETW6nExIK5q
4b9ls9BzTeGjPI3mFoHNlt CHwXL8qA4VgMSkTHIVkO3K90oagS/mYxfFng4DuOlITqEPd5LftZKLJB QzD/7ykIVPFMQmfmH
xEjKii+vkcxQN0SF9kB
example.com.          172800  IN      SIG     KEY 3 2 172800 20030202164237 20030103164237 61511 co
m. CIa0z6RSwqfEJY+SSqH/zWS7cSXxHrXI2wCGPlErA0MIsLGuciXsZP4=

;; Query time: 2 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Jan  5 18:55:41 2003
;; MSG SIZE  rcvd: 961
```

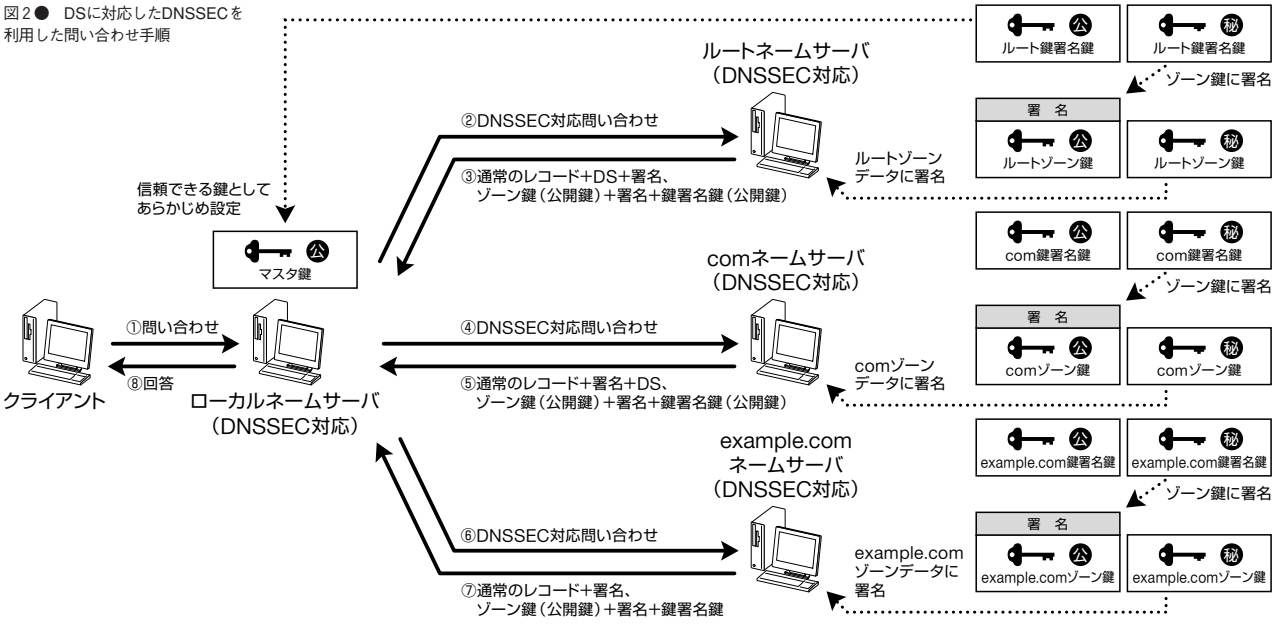
リスト6● digコマンドによる問い合わせの結果（リソースレコードが存在しなかった場合）

```
$ dig @ns1.example.com. +dnssec +norec sykes.example.com.

; <<>> DiG 9.2.1 <<>> +dnssec +norec sykes.example.com.
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 18567
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;sykes.example.com.          IN      A

;; AUTHORITY SECTION:
example.com.          3600    IN      SOA     ns1.example.com. hostmaster.example.com. 2003021801 2
8800 7200 604800 3600
example.com.          3600    IN      SIG     SOA 3 2 86400 20030204094924 20030105094924 49073 exa
mple.com. CGVIoxjgmCEiSiZyJm91364qt6plEymh9uQNxZaWpD7cORLL3PcGAwU=
ns2.example.com.      86400   IN      NXT     www.example.com. A SIG NXT
ns2.example.com.      86400   IN      SIG     NXT 3 3 86400 20030204094924 20030105094924 49073 exa
mple.com. CDQQr2A+zlD1kw5hd2XjFSo5BvcSdLbtK/prLabfThKvRkgoQio1R1I=

;; ADDITIONAL SECTION:
example.com.          172800  IN      KEY     256 3 3 CPQ4z1GPtlwH/X90MO0HEsRYR/gfgwicOjK2niyWFilsi
vyMlZSV3iZ/ SFAHhXPG4BYcCi71eZg7X6JLEVv5rKBjVS5IbacXCO5qJUY6RkoJVLJV mZZYef12PCsB/fMXQ1CADR/uTTmP
qeuj8m0lK59c4L3nD16okiyF1JHi 1Tsk7t1BPtsZ61h6Ks37p+aueg/iJ+1325Rryr6qC/7iUlXvLaYAxy7q a1TRwOvpYoX
xotQ37Xs4eUIQ3KfCYQwPaW0fWe0EvtanpkkDlKUwuWPo NwrLvmUF3OAwyleLEQiCxepPYmp3EWTlUh6tIVo9ygj4GF3yk7+
IOFbm PGrjmgQmC0TSpMwdukEzZBlfaCI/u3MUdSgX+uslYw/3tpORlrMP9muh DiUlSWe5MURREXlS9/OFTFqETW6nExIK5q
4b9ls9BzTeGjPI3mFoHNlt CHwXL8qA4VgMSkTHIVkO3K90oagS/mYxfFng4DuOlITqEPd5LftZKLJB QzD/7ykIVPFMQmfmH
xEjKii+vkcxQN0SF9kB
example.com.          172800  IN      SIG     KEY 3 2 172800 20030202164237 20030103164237 61511 co
```

リスト6続き

```
m. CIa0z6RSwqfEJY+SSqH/zWS7cSXxHrXI2wCGPlErA0MIsLGuciXsZP4=

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Jan  5 19:07:10 2003
;; MSG SIZE  rcvd: 799
```

図2● DSに対応したDNSSECを利用した問い合わせ手順




```
sub.example.com.  86400 IN  NS  ns.sub.example.com.
                  86400 IN  DS  28668 3 1 49FD46E6C4B45C55D4AC69CBD3CD34AC1AFE51DE
```

リスト7● 鍵タグが28668である公開鍵を下位ゾーンの鍵署名鍵として承認するためのDSリソースレコード

<owner> <ttl> <class> DS <key tag> <algorithm> <digest type> <digest>

<key tag>には、下位ゾーンで使用する鍵署名鍵から計算された鍵タグが入り、<algorithm>にはその鍵が使用する署名アルゴリズムの番号が入る。また、<digest type>には、SHA-1などの公開鍵のダイジェストを生成するためのハッシュ関数の番号が入り、<digest>には、下位ゾーンで使用する鍵署名鍵のダイジェストが入る。

例えば、鍵タグが「28668」である公開鍵を下位ゾーンの鍵署名鍵として承認するためには、リスト7のようなDSリソースレコードを記述する。

今回はDNSSECの仕組みについて説明した。次回は、多言語ドメインについて解説する。

NTTデータ　馬場達也

●今回の内容に関連するRFC

RFC 2535 "Domain Name System Security Extensions"
RFC 2536 "DSA KEYs and SIGs in the Domain Name System (DNS)"
RFC 2537 "RSA/MD5 KEYs and SIGs in the Domain Name System (DNS)"
RFC 2539 "Storage of Diffie-Hellman Keys in the Domain Name System (DNS)"
RFC 3008 "Domain Name System Security (DNSSEC) Signing Authority"
RFC 3090 "DNS Security Extension Clarification on Zone Status"
RFC 3110 "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)"
RFC 3225 "Indicating Resolver Support of DNSSEC"
RFC 3445 "Limiting the Scope of the KEY Resource Record out"