

Internet Week 2010 S10
DNSSECチュートリアル
～実践編～

民田雅人 <minmin@jprs.co.jp>
株式会社日本レジストリサービス

目次

- DNSキャッシュへの毒入れ
- DNSSECのしくみ
- DNSSEC導入に向けて
- DNSSECの鍵と信頼の連鎖
- DNSSECのリソースレコード(RR)
- 鍵更新と再署名
- BINDキャッシュサーバでのDNSSECの設定
- 鍵生成と署名作業
- BIND権威サーバでのDNSSECの設定
- スマート署名 (Smart signing)
- 全自動ゾーン署名
- DNSSEC化によるDNSデータの変化
- DNSSECのリスク
- まとめ

DNSキャッシュへの毒入れ

DNSプロトコル

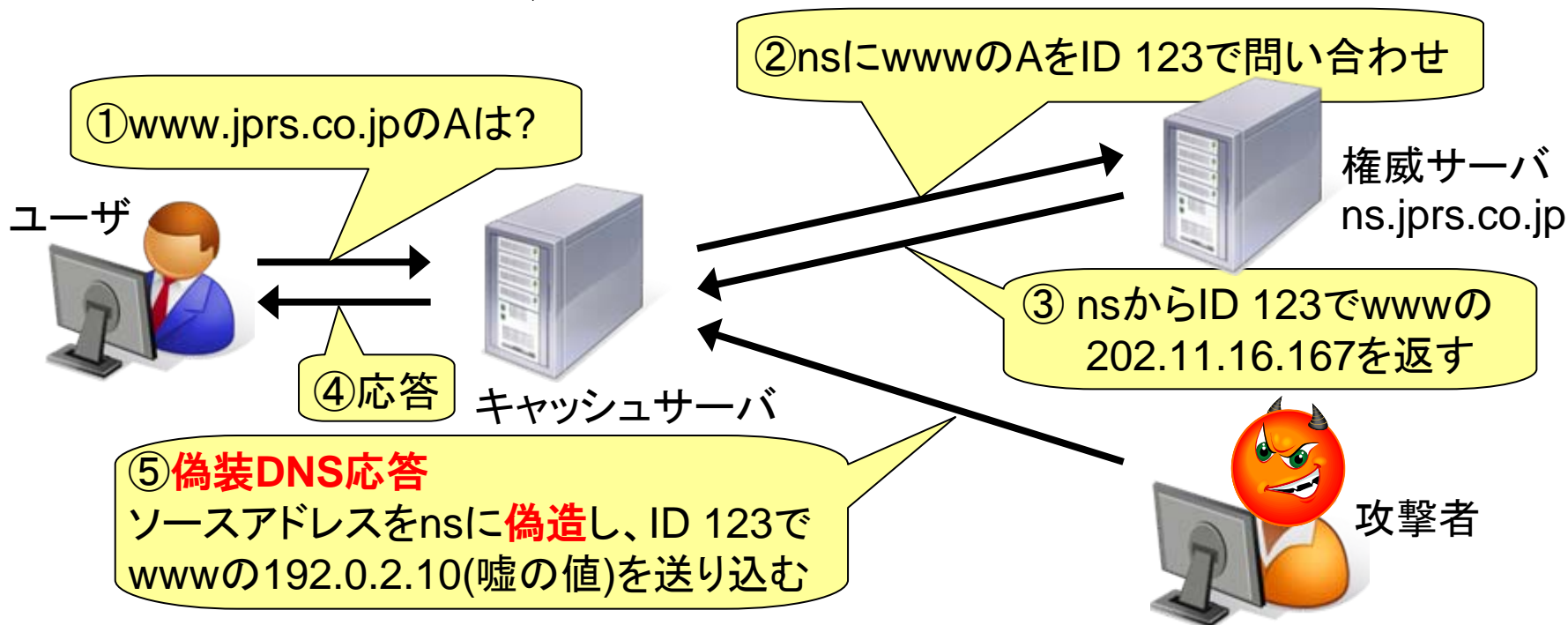
- DNSの通信は問合せと応答の単純な往復
 - この名前のIPアドレスは? ⇒ IPアドレスはXXXXだよ
- トランスポートは主にUDP
 - 問合せパケット クエリ名 + ID + etc...
 - 応答パケット クエリ名 + ID + 応答 + etc...

ID: 識別のための16bitの値
- キャッシュDNSサーバは、応答パケットを、
ソースアドレスとポート、クエリ情報、ID、で識別
- 条件が一致すれば応答パケットを正しいと判断

偽装応答型による毒入れ

- なんらかの手段を使い、本来の応答より先に偽装応答をキャッシュサーバに送り込み、偽情報をキャッシュさせる手法
 - 通常時DNSはUDPで通信するため、偽装応答が容易
- 攻撃手法
 - キャッシュサーバのDNS検索を盗聴し偽装応答を返す
 - キャッシュサーバに問い合わせを送り、IDを変化させた複数の偽装応答を返す(オープンリゾルバは非常に危険)
 - TTLの短いレコードを狙って、キャッシュサーバに偽装応答を送り続ける
 - etc...

偽装応答型の攻撃



- ③より先に⑤の偽DNS応答が送り込まれると、キャッシュサーバは嘘情報をキャッシュする
- ④で嘘情報をクライアントに送り、クライアントは偽のサイトへ誘導される

偽装応答型の攻撃が成功する確率

問い合わせと応答のIDが一致すれば攻撃が成功
攻撃1回あたりの成功確率

$$P_s = \frac{R \times W}{N \times Port \times ID}$$

R: 攻撃対象1台あたりに送るパケット量(pps)

W: 攻撃可能な時間(Query⇒AnswerのRTT)

N: 攻撃対象レコードを保持する権威サーバの数

Port: キャッシュサーバのQuery portの数

ID: DNSのID (16bit = 65536)

(*R* 20000pps, *W* 10ms, *N* 2, *Port* 1で 0.00152)

毒入れ攻撃 まとめ

- 偽装応答型の攻撃成功確率は決して低くない
 - 現実的には、攻撃に失敗するとキャッシュDNSサーバは正規のレコードをキャッシュするため、連続攻撃は不可能
- 画期的攻撃手法であるKaminsky型の攻撃手法は、連続攻撃が可能 ⇒ ほぼ確実に攻撃が成功する
- 毒入れは、DNSプロトコルそのものが持つぜい弱性
 - UDPを使う、IDが16bitしかない、etc...
- 毒入れからDNSキャッシュを守るためのセキュリティ面でのプロトコル拡張が**DNSSEC**

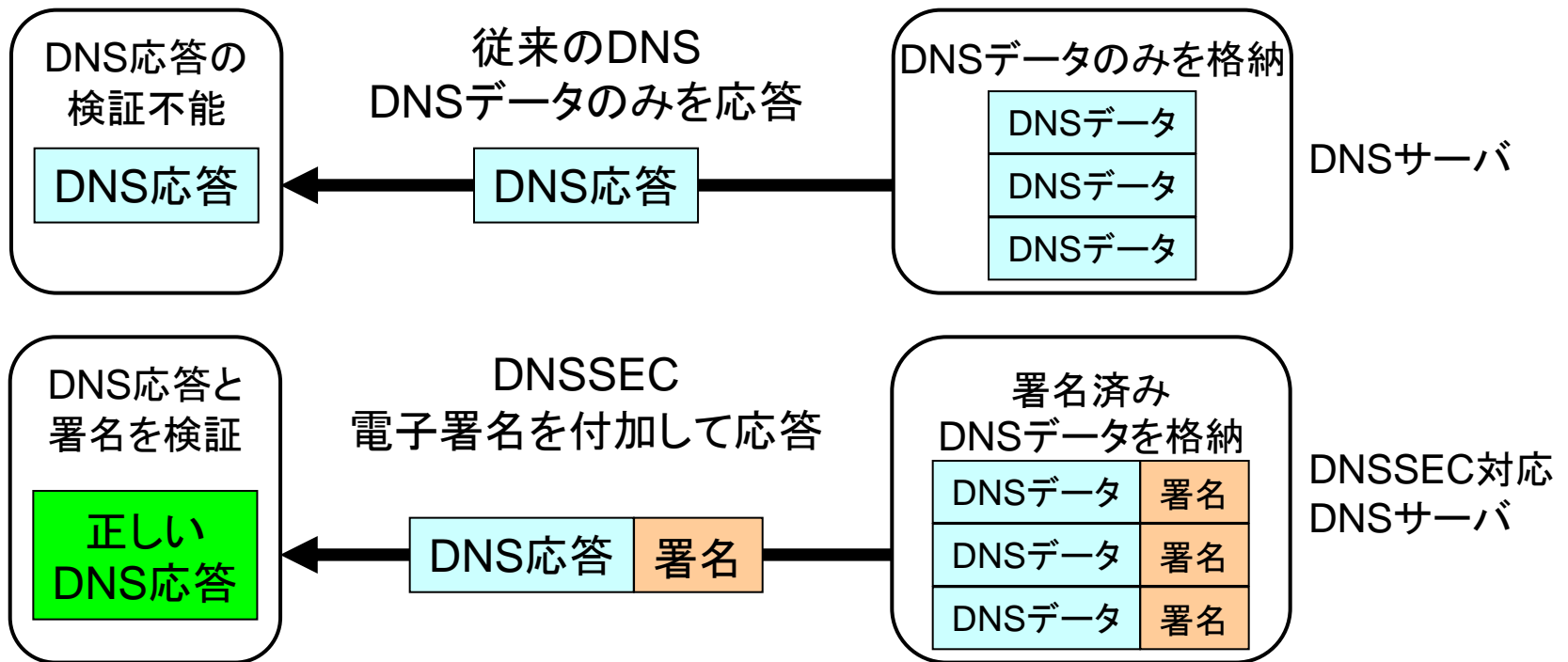
DNSSECのしくみ

DNSSECとは

- DNSセキュリティ拡張(DNS SECurity Extensions)
 - **公開鍵暗号**の技術を使い、検索側が受け取ったDNSレコードの出自・完全性(改ざんのないこと)を検証できる仕組み
 - 従来のDNSとの**互換性を維持した拡張**
 - Kaminsky型攻撃手法の発覚を1つの契機に、多くのTLDが導入開始あるいは導入予定
- キャッシュへの毒入れを防ぐことができる現実解
 - 他の技術も存在するが標準化が成されていない

従来のDNS vs DNSSEC

- DNSサーバが応答に電子署名を付加し出自を保証
- 問合せ側でDNS応答の改ざんの有無を検出できる

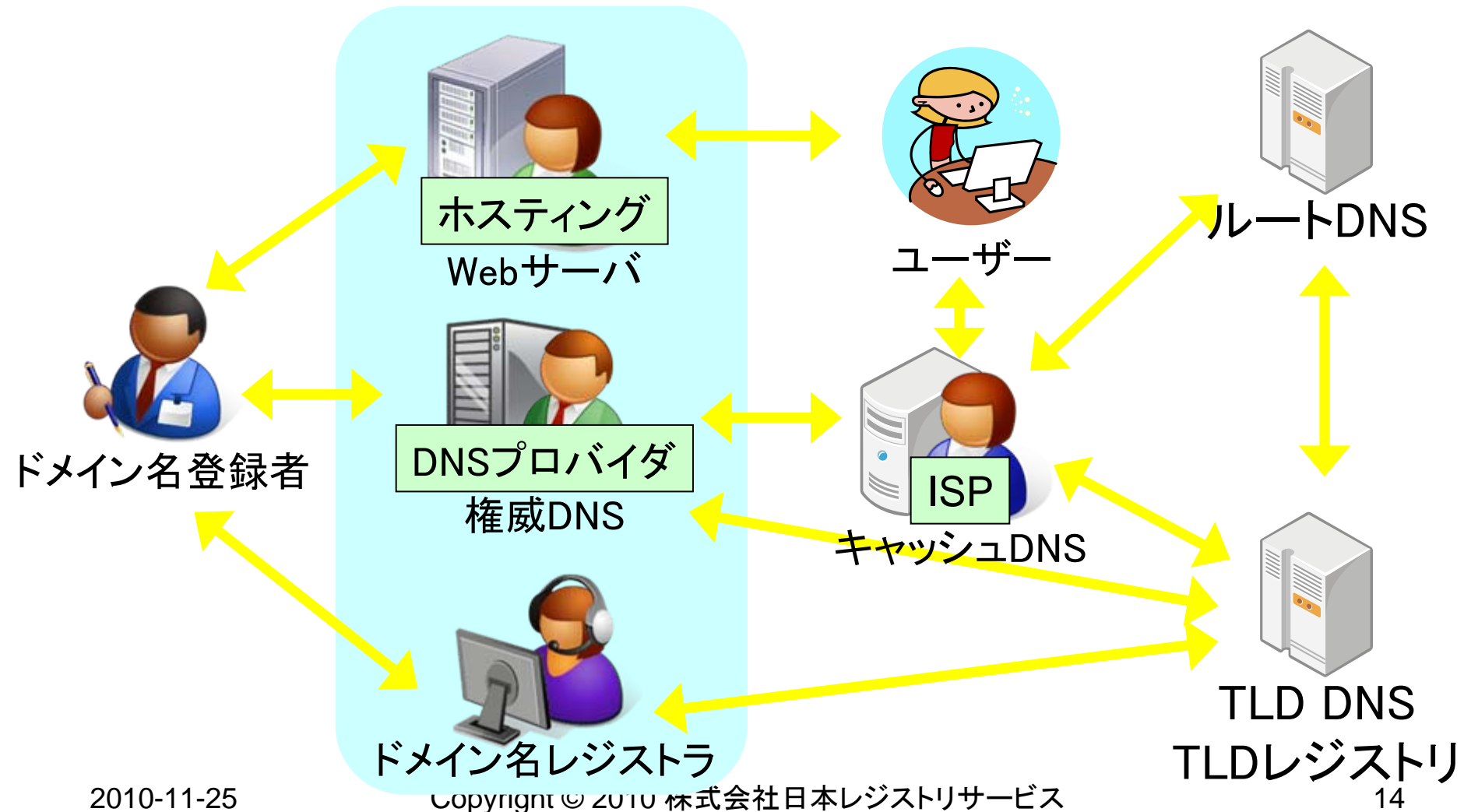


DNSSECのスコープ

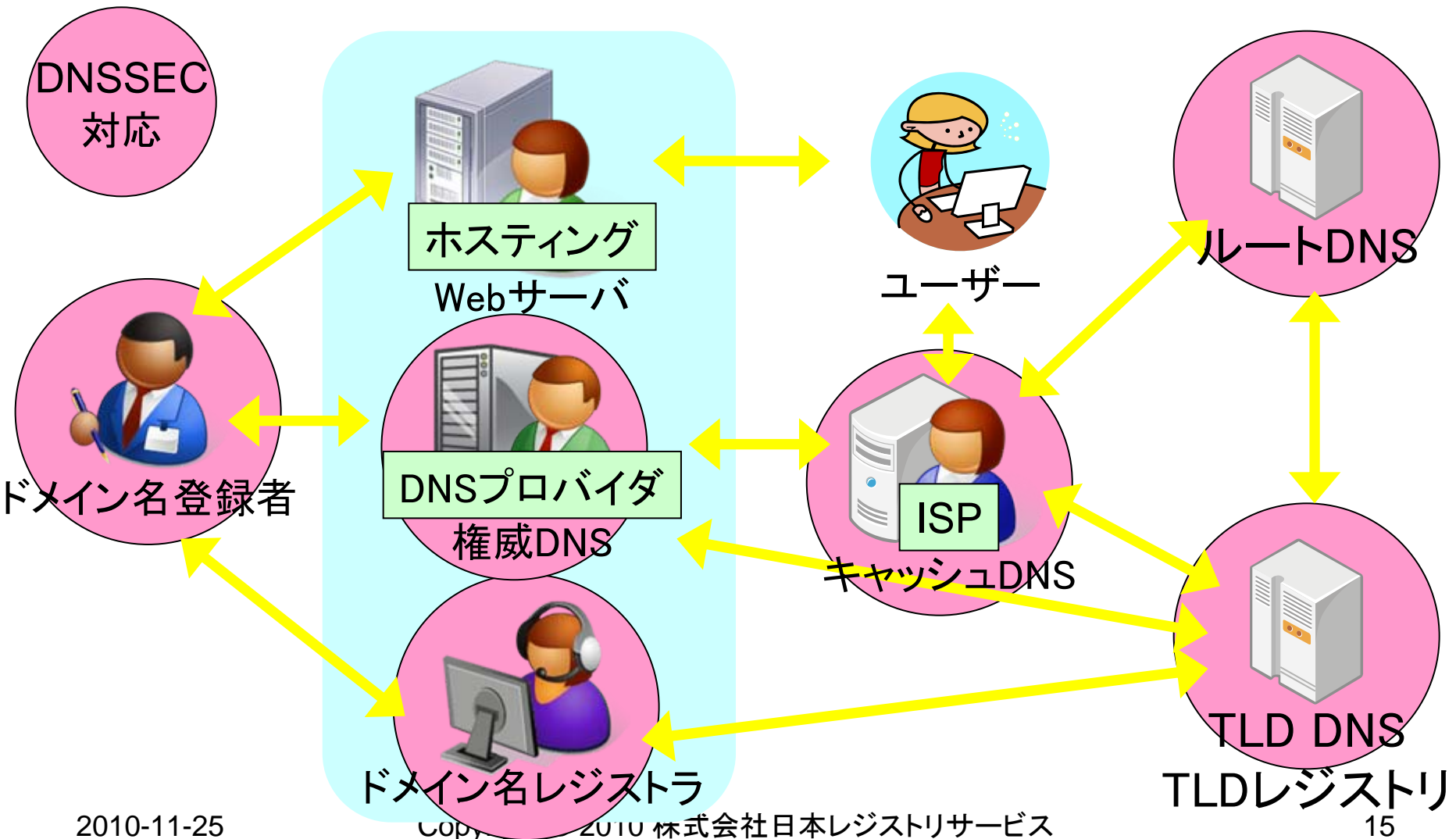
- 対象としているもの
 - DNS問合せの応答が、ドメイン名の正当な管理者からのものであることの確認
⇒ **出自の保証**
 - DNS問合せの応答における、DNSレコードの改変の検出
⇒ **完全性の保証**
- 対象としていないもの
 - 通信路におけるDNS問合せと応答の暗号化
※DNSレコードは公開情報という考え方から

DNSSEC導入に向けて

DNS関係者と各情報の流れ



DNSSEC対応が必要な関係者



DNSSEC対応作業の概要

- ドメイン名登録者
 - DNSSEC導入の決定
- ドメイン名レジストラ
 - 鍵情報の上位レジストリへの取次ぎ
- TLD DNS、ルートDNS
 - 権威DNSサーバのDNSSEC対応化
 - ゾーンへの署名
- DNSプロバイダ
 - 権威DNSサーバのDNSSEC対応化
 - 秘密鍵・公開鍵を作成し、ゾーンに署名
- ISP
 - キャッシュDNSサーバのDNSSEC対応化
 - (キャッシュDNSサーバでの)署名の検証

世界のDNSSEC導入の概況

(2010年11月8日現在)

- rootゾーン
 - 2010年7月15日よりDNSSECの正式運用開始
- DNSSEC導入済TLD
 - rootゾーンにある全294のTLDのうち
 - 62のTLDが署名済み
 - 46のTLDがrootゾーンにDSを登録済み
 - (2009年末は10のTLDが署名済みだったのみ)
- 今後の状況
 - jpは2010年10月17日に署名開始、2011年1月16日より登録受付サービス開始
 - com は2011年前半 / netは2010年末に導入予定
 - 導入予定のTLDは多数

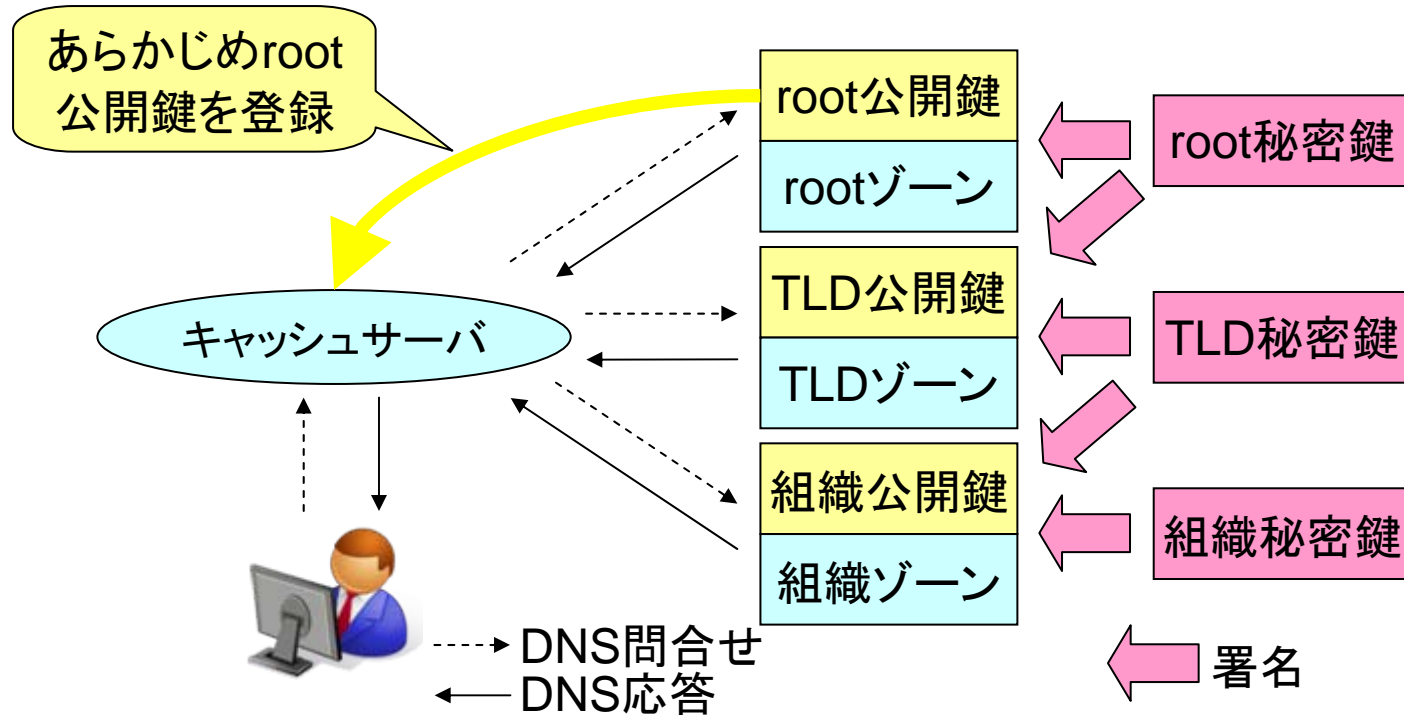
DNSSEC普及に関連した動き

KIDNS (Keys In DNS)

- DNSで証明書を公開するアイデア
 - CERT RR (RFC 4398 2006年 Obsoletes RFC 2538)
- DNSSECの実用化にともない、現在実用化の検討が始まっている
 - DNSSECによってDNSレコードが信用できる
⇒ DNSにある証明書も信用できる
 - 従来の自己証明書では、本当にそのサイトのものかどうか確認が困難だが、DNSはそのサイトのもの
 - ドメイン名の一致を重要な目的とする証明書では、DNSで自己署名証明書を配布するほうがリーズナブル?

DNSSECの鍵と信頼の連鎖

DNSSECの信頼の連鎖の概念図



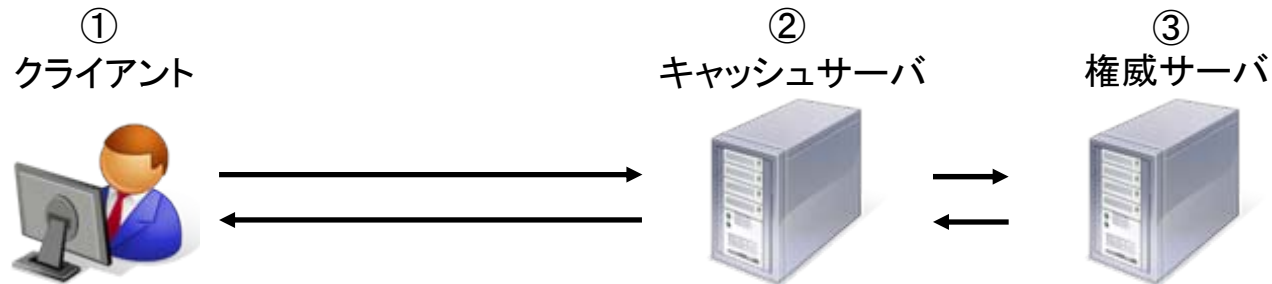
- 秘密鍵で、自ゾーンと下位ゾーンの公開鍵に署名
- root公開鍵をキャッシュサーバに登録することで、rootから組織ゾーンまでの信頼の連鎖を確立

用語：バリデータ(Validator)

- DNSSECにおいて、バリデータは署名の検証を行うもの(プログラム、ライブラリ)を指す
- バリデータの所在
 - キャッシュサーバが署名検証を行う場合、キャッシュサーバがバリデータそのもの
⇒ 現状、もっとも一般的なDNSSECのモデル
 - WEBブラウザ等のDNS検索を行うアプリケーションが直接署名検証を行うモデルも考えられる

DNSSEC化による 名前解決モデルの変化

- 従来のDNSでの名前解決モデル



- DNSSECでの名前解決モデル



– 多くの場合バリデータは②に実装

– バリデータが①に実装されていても問題ない

DNSSEC利用する2種類の鍵とDS

- 2種類の鍵
 - ZSK (Zone Signing Key)
ゾーンに署名するための鍵
 - KSK (Key Signing Key)
ゾーン内の公開鍵情報に署名するための鍵
- DS (Delegation Signer)
 - 上位ゾーンに登録するKSKと等価な情報

ZSK

- 比較的暗号強度の低い鍵
 - 例えばRSAで1024bit等の鍵を使う
- 暗号強度が低い
 - 署名コストが低いため、大規模ゾーンの署名にも適応できる
 - 安全確保のため、ある程度頻繁に鍵を更新する必要がある
- 鍵更新は親ゾーンとは関係なく独立で行える

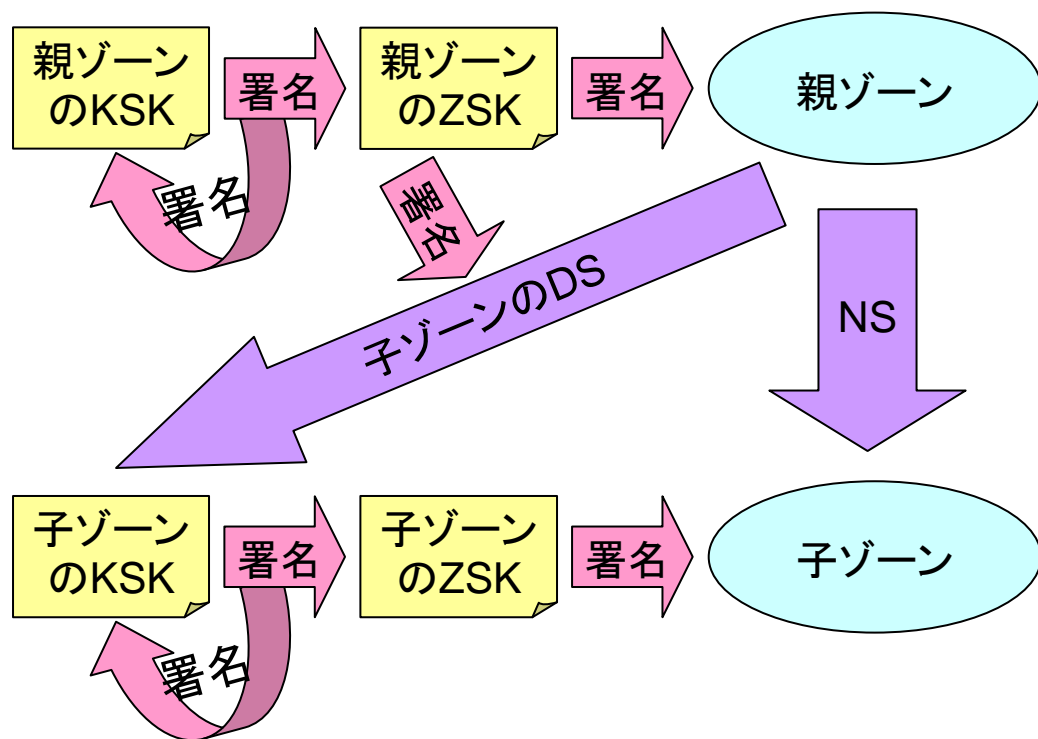
KSK

- 比較的暗号強度の高い鍵
 - 例えばRSAで2048bitの鍵を使う
- 暗号強度が高い
 - 利用期間を長くできるため、鍵更新の頻度を低くできる
 - 署名コストは高いが、少数の鍵情報のみを署名対象とするため問題にはならない
- KSK公開鍵と暗号論的に等価な情報(DS)を作成し、親ゾーンに登録する
 - **KSKを変更する場合、同時にDSも更新する**

DS

- KSK公開鍵を、SHA-1/SHA-256等のハッシュ関数で変換したDNSレコード
⇒ KSK公開鍵と等価の情報
- 親ゾーンの委任ポイントに、NSと共に子ゾーンのDS情報を登録
 - 親ゾーンの鍵でDSに署名してもらうことで、信頼の連鎖を形成する

DNSSECの信頼の連鎖



- 公開鍵暗号による信頼の連鎖を形成
- キャッシュサーバが、KSKの公開鍵を使って署名を検証
⇒ **トラストアンカー**
 - キャッシュサーバにはrootゾーンのKSK公開鍵を登録する

DSとNSの本質的な違い

- NS: 委任先DNSゾーンデータが存在する(可能性のある)**サーバを指し示す**
- DS: 委任先**DNSゾーンデータを直接指し示す**
 - DSは子ゾーンのKSKと等価な情報
- NSの指し示すドメイン名がDNSSEC非対応であってもDNSSECの検証は問題無い

jpゾーンでの例

```
example.jp. IN NS  ns0.example.ad.jp.  
example.jp. IN DS  2260 8 2 CC83B074566.....
```

- example.ad.jpドメイン名はDNSSEC対応していなくても、example.jpドメイン名はDNSSEC検証可能

DNSSECの リソースレコード(RR)

DNSSEC関係のRR一覧

- DNSKEY KSK・ZSK公開鍵の情報
- RRSIG 各RRsetへの署名
- DS KSK公開鍵のハッシュ値を含む情報(親ゾーンに登録)
- NSEC 次のRRへのポインタと存在するレコード型の情報
- NSEC3 NSECを改良したもの(後述)
- NSEC3PARAM NSEC3に必要な情報

DNSKEY RR

- KSKとZSKの公開鍵を示すRR
 - オーナー名はゾーン頂点(=ゾーン名)
 - KSKとZSKを必要に応じて複数(後述)設定

```
example.jp. IN DNSKEY
  256 3 5 AwEAAeNO41ymz+Iw(行末まで省略)
  ① ② ③ ④
```

- ① フラグ(256:ZSK、257:KSK)
- ② プロトコル番号 (3のみ)
- ③ DNSSECアルゴリズム番号
- ④ 公開鍵(Base64で符号化)

DNSSECアルゴリズム番号(抜粋)

番号	略称	参照
5	RSASHA1	[RFC3755] [RFC3110]
7	NSEC3RSASHA1	[RFC5155]
8	RSASHA256	[RFC5702]
10	RSASHA512	[RFC5702]

注) 5と7に差は無く、NSECとNSEC3 (後述) で使い分ける

DNSSECのDNSSECアルゴリズム番号一覧

<http://www.iana.org/assignments/dns-sec-alg-numbers>

DS RR

- DS - Delegation Signer
 - 子ゾーンのKSKの正当性を親ゾーンで承認
 - 親ゾーンにのみ記述する唯一のRR

```
example.jp. IN DS 63604 5 1 DF...(16進数40文字)  
example.jp. IN DS 63604 5 2 E8...(16進数64文字)  
                          ①      ②  ③                                  ④
```

- ① 鍵のID
- ② DNSSECアルゴリズム番号
- ③ ハッシュのアルゴリズム(1:SHA-1, 2:SHA-256)
- ④ ハッシュ化したKSK公開鍵

RRSIG RR

- 各RRへの署名で、RRset毎に存在する

```
ns.example.jp. IN RRSIG A 5 3 86400  
                        ① ② ③ ④  
                20091208144031 20091108144031 40002 example.jp.  
                        ⑤ ⑥ ⑦ ⑧  
                NiVihYAIZBEwfUUAbPazDRiBvhNH8S(以下省略)  
                        ⑨
```

- ① 署名対象のRRの種類

ここではns.example.jpのA RR

- ② DNSSECアルゴリズム番号

- ③ ラベルの数

”ns.example.jp”だと3、”*.example.jp”だと2

RRSIG RR(続き)

- ④ 署名対象RRのTTL
- ⑤ 署名の有効期限
- ⑥ 署名の開始時刻
- ⑦ 鍵のID
- ⑧ ドメイン名
- ⑨ 署名

署名は、元のRRの全て(TTL、クラス等を含む)と、RRSIGの署名そのものを除いた残りを含めて計算する

DNSSECにおける不在証明

- DNSSECではドメイン名が存在しない場合、**存在しないことを証明**する必要がある
 - 万が一存在しないドメイン名を偽装されたときのために、存在するのかどうかを検証できる仕組みが必須
- 存在するRRは署名(RRSIG RR)を付加して検証することで存在を証明できる
 - ⇒ 存在しないRRは署名不能

ハンバーガーのパティの有無

- パティ(肉)が有る
 - パティの存在を判断できる
- パティが無く、バンズ(パン)も無い
 - パティが無いかどうか判断不能
 - ⇒ 単純に配膳が遅れているだけ?
- クラウン(バンズ上部)とヒール(バンズ下部)があるのにパティが無い
 - クラウンとヒールの存在が判断できる
 - ⇒ パティが存在しないことを確実に判断できる



NSEC RR

- NSECは存在しないものを証明(署名検証)するためのRR
 - 存在するレコードすべてを整列し、次のレコードへのリストを生成することで、存在しないものを証明する
 - NSEC RRにRRSIGを付加し署名検証を行う

NSEC RRの例

- sec2.example.jpを問合せた場合の応答

```
sec1.example.jp.  IN  NSEC  
sec3.example.jp. NS DS RRSIG NSEC
```

(権威セクションで応答)

- sec1.example.jp の次(アルファベット順)のドメイン名は sec3.example.jp
⇒ sec2.example.jp は存在しないことを示す
- sec1.example.jp には NS, DS, RRSIG, NSEC のRRが存在する
⇒ NSECはRR TYPEの不存在も証明する

NSEC3 RR

- NSECを使った不在証明では、NSEC RRを辿れば、完全なゾーンデータを手に入れる
⇒NSEC方式はゾーンデータの公開と等価
 - Walker(DNSSEC Walker)というツールで、NSEC方式のDNSSEC化ゾーンのデータを手に入れ可能
- **NSEC3** (RFC 5155)
 - ドメイン名を一方方向性ハッシュ関数でハッシュ化したものを整列する

NSEC3 RRの例

```
4HTJTU7UP56274L1C00Q9MLPHG2A2H85.example.jp.  
IN NSEC3  
1 0 3 123ABC ←NSEC3の関連パラメータ  
B0B790UE4SAE4QB4RTB3PJSIH6JAOB7R NS DS RRSIG
```

NSEC RRと比べると

- ラベルがハッシュ化されBase32でエンコード
 - 元のドメイン名は推測不能
- NSEC3の関連パラメータを付加

NSEC3の関連パラメータ

- 前スライドのRR例

1 0 3 123ABC
① ② ③ ④

- ① ハッシュアルゴリズム(1:SHA-1 RFC5155)
- ② NSEC3 オプトアウトフラグ
(1ならオプトアウト、0はオプトアウトしていない)
- ③ 繰り返し
- ④ ソルト(16進数で表記。例は3バイト分のソルト)

NSEC3のハッシュ値計算方法

- ハッシュの計算アルゴリズム
 - ① 値にソルトを結合
 - ② 次にハッシュアルゴリズムでハッシュ値を計算
 - ③ ①、②を繰り返して指定された回数適用する
- 計算の元になる値は、小文字で正規化したドメイン名(のワイヤーフォーマット)
 - nsec3hash (BIND 9.7系以降に付属)で計算可

NSEC3でのオプトアウト

- 一部の委任先がDNSSEC化している場合
 - 主に.JP、.COM等のTLDで該当
- ゾーン内のレコード全てにNSEC3 RRを用意すると、それに付随するRRSIGを含めた計算コストが膨大となる
 - DNSSEC化していない委任情報に、署名を付加する必要性は薄い
- 必要のある委任先にのみNSEC3 RRを用意

NSEC3PARAM RR

```
example.jp. IN NSEC3PARAM 1 0 3 123ABC
```

- ゾーン提供(権威サーバ)側が、NSEC3の計算を行うために必要なレコード
 - NSEC3のパラメータを抜き出したもの
 - オーナー名はゾーン頂点(ゾーン名)

NSEC3での権威サーバの応答

- 存在しない名前の検索を受けた権威サーバ
 - クエリ名のハッシュ値を計算
 - あらかじめ整列してあるNSEC3 RRの中から、前後に該当するものを署名と共に権威セクションで応答
 - 実際は複数のNSEC3を応答
(説明省略 RFC5155 Section 7.2参照)
- NSEC3のオーナー名の間合せ
 - ドメイン名としては存在しないもの
⇒ 名前エラー(Name Error)を応答する

NSECとNSEC3比較

	NSEC	NSEC3
ゾーンデータの秘匿性	無	有
ハッシュ値を求めるための 計算コストの増加	無	有

- NSEC3はNSECに比較しドメイン名の秘匿性は高まるが、ハッシュ計算のためのコストが増加する
⇒ NSEC3とNSECは用途に応じて使い分ける
 - ゾーンデータを秘匿する必要が無い場合、NSECのほうが各DNSサーバの負荷の増加を抑えられる

www.example.jpのAの署名検証 (1)

- ① 上位からNSとDSを受け取る
 - JPの権威サーバからexample.jpのDS (DSの署名検証の解説は省略)とNSの情報を受け取る
- ② 当該ゾーンのDNSKEYを受け取る
 - example.jpの権威サーバから、example.jpのDNSKEY(複数)とRRSIG(複数)を受け取る
- ③ DNSKEYからKSKを識別する
 - DNSKEYは複数(2個以上)存在するので、フラグが257のDNSKEY(1個以上)を識別する
- ④ KSKを特定する
 - KSKとDSの鍵ID、DNSSECアルゴリズム番号を比べ、KSKを特定

www.example.jpのAの署名検証 (2)

⑤ KSKを認証する

- DSのハッシュアルゴリズムに従ってKSKのハッシュ値を計算し、DSにあるハッシュ値と比較してKSKを確認する

⑥ DNSKEYを認証する

- ③で受け取ったDNSKEYに付随したRRSIG(複数)の鍵IDからKSKの鍵IDと一致するものを識別し、署名検証を行いDNSKEYを認証する

⑦ DNSKEYからZSKを識別する

- DNSKEYのフラグが256のものを識別する
ここでZSKは複数存在する可能性がある

www.example.jpのAの署名検証 (3)

- ⑧ www.example.jpのAを受け取る
 - example.jpの権威サーバから、AとRRSIG(1個以上)を受け取る
- ⑨ www.example.jpのAを認証する
 - RRSIGの鍵IDと一致するZSKで署名を検証する
 - 必ずしも処理はこの順番ではなく、実装に依存する
 - 署名検証の際、署名の有効期間、ドメイン名など他のRRSIGのパラメータもチェックされる
 - DSやDNSKEY、RRSIG等は、署名検証後もTTLの有効時間キャッシュする

鍵更新と再署名

鍵更新

- 鍵更新: Key rollover
 - 同じ鍵を長期間使い続けると、様々なリスクが生じる
 - 不注意、偶発的事故、鍵の盗難、暗号解読等
- リスクを最小に抑えるため、DNSSEC対応ゾーンの運用では定期的な鍵更新(鍵の交換)を行う
 - 例えばSE(スウェーデン)の場合、年に1回新しいKSKを生成し、2年間利用する運用を行っている

鍵更新時に留意すべきこと

- 鍵更新は、DNSSECの信頼の連鎖が途切れないよう、注意深く作業する必要がある
 - 鍵情報(DSやDNSKEY)と署名(RRSIG)はDNSのレコードである
 - ⇒ キャッシュサーバはこれらを**キャッシュ**する
- キャッシュしている情報と、あらたにキャッシュサーバが受け取る情報の整合性を確保する
- 2種類の鍵更新手法
 - 事前公開法 (Pre-Publish Key Rollover)
 - 二重署名法 (Double Signature Key Rollover)

ZSKの更新

事前公開法 (1/2)

- ① DNSKEYに新旧のZSKを登録する
 - 新ZSKを作成し、旧ZSKと共にDNSKEYに登録し(この状態でKSKを含めてDNSKEYは最低3個)、旧ZSKでゾーンを署名
 - DNSKEYのTTL時間(+セカンダリの転送時間)待つ
 - 全てのキャッシュサーバが新旧のZSKを含んだDNSKEYをキャッシュするようになり、旧RRSIGでも新RRSIGでも署名を検証できるようになる
- ② ゾーンの署名鍵を新ZSKに切り替える
 - ゾーン内の最長のTTL時間(+セカンダリの転送時間)待つ
 - 全てのキャッシュサーバから旧ZSKで署名したRRSIGが無くなる

ZSKの更新 事前公開法 (2/2)

③ 旧ZSKをDNSKEYから削除する

– DNSKEYは新ZSKとKSKの状態になる

	初期状態	①	②	③
DNSKEY	KSK 旧ZSK	KSK 旧ZSK 新ZSK	KSK 旧ZSK 新ZSK	KSK 新ZSK
RRSIG	旧ZSKでの 署名	旧ZSKでの 署名	新ZSKでの 署名	新ZSKでの 署名



ZSKの更新 二重署名法

- ① 新ZSKを作成し、新旧両方のZSKでゾーンを署名
– ゾーン内の最大TTL時間(+セカンダリの転送時間)待つ
- ② DNSKEYのZSKの新旧を入れ替え、新ZSKでゾーンを署名

	初期状態	①	②
DNSKEY	KSK 旧ZSK	KSK 旧ZSK	KSK 新ZSK
RRSIG	旧ZSKでの 署名	旧ZSKでの署名 新ZSKでの署名	新ZSKでの 署名



ZSKの更新 メリット・デメリット

- 事前公開法
 - ゾーンへの署名を2回行う必要が無い
 - × ZSKの公開時間が長くなるため、暗号解読攻撃のリスクが高まる(ZSKは鍵長が短い)
 - △ 初期状態から数えて4ステップ必要
 - 次のZSKを常時公開することで、手順を簡略化可能
- 二重署名法
 - 初期状態から数えて3ステップで終了する
 - × ゾーンへの署名を2回行う必要がある
 - × 鍵更新期間中(①の状態)はDNSデータが大きくなる

KSKの更新

二重署名法(1/2)

- ① 新KSKを作成し、DNSKEYに登録して、DNSKEYを新KSKと旧KSKで署名する
- ② 親ゾーンのDS登録を旧から新に切り替える
 - 親側のDSの切り替え作業を待ち、その後親側の旧DSのTTL時間分待つ
- ③ 旧KSKを削除する

KSKの更新 二重署名法(2/2)

	初期状態	①	②	③
親ゾーンのDS	旧DS	旧DS	新DS	新DS
子ゾーン DNSKEY	旧KSK ZSK	旧KSK 新KSK ZSK	旧KSK 新KSK ZSK	新KSK ZSK
子ゾーン DNSKEYの RRSIG	旧KSKでの 署名 ZSKでの 署名	旧KSKでの 署名 新KSKでの 署名 ZSKでの 署名	旧KSKでの 署名 新KSKでの 署名 ZSKでの 署名	新KSKでの 署名 ZSKでの 署名

時間の流れ



KSKの更新 事前公開法

- ① 新KSK(と新DS)を作成し親ゾーンに新旧2つのDSを登録する
 - 親ゾーンのDS登録を待つ、さらに旧DSのTTL時間待つ
- ② 旧KSKを破棄し、新KSKでDNSKEYに署名する
- ③ 親ゾーンのDS登録を新DSのみにする

KSKの更新 事前公開法

	初期状態	①	②	③
親ゾーンのDS	旧DS	旧DS 新DS	旧DS 新DS	新DS
子ゾーン DNSKEY	旧KSK ZSK	旧KSK ZSK	新KSK ZSK	新KSK ZSK
子ゾーン DNSKEYの RRSIG	旧KSKでの 署名 ZSKでの 署名	旧KSKでの 署名 ZSKでの 署名	新KSKでの 署名 ZSKでの 署名	新KSKでの 署名 ZSKでの 署名



KSKの更新

メリット・デメリット

- 二重署名法
 - 親ゾーンとのDSのやり取りが1回で済む
 - ZSKと違い、署名がDNSKEYにのみ作用するので、ゾーンデータの肥大化は問題にならない
- 事前公開法
 - 親ゾーンとDSのやり取りが2回必要となる

ゾーンの再署名

- 署名の有効期限が長すぎるのは望ましくない
 - 万が一の事態(鍵の盗難等)において速やかに対応するためには、署名期間は短いほうがよい
- 有効期限が数分の署名も技術的には可能
 - 休日等の対応を考慮すると現実性に欠ける
- 署名の有効期限に達する前に署名の有効期限を更新するために、**ゾーン全体の再署名**が必要となる
 - DNSSECでは、再署名を行ってゾーン情報を定期的に更新する必要がある

NSEC3固有の問題

- NSEC3では、同じハッシュ値を使い続けると辞書攻撃により秘匿している情報が解析されるリスクがある
 - ゾーンの再署名時にソルトを変更し、ハッシュ値を変えるのが望ましい

鍵の更新間隔

- 運用面での鍵の更新間隔の実用的な値
 - KSK 13カ月 12ヶ月で鍵更新
 - ZSK ~3カ月
- KSKの更新はDSの登録変更作業を伴うため、ドメイン名登録の更新にあわせるのが現実的
- ZSKはKSKのような制約は無く、ゾーン内で処理が完結するため、運用面での負荷を考慮しながら期間を短めに設定する

TTLと署名の期間

- RRのTTLが署名期間より長かったら
 - キャッシュしたRRの署名が無効になる事態が発生する
 - ⇒ 署名の有効期間はTTLより長い必要がある
- SOAのExpireが署名期間より長かったら
 - セカンダリサーバでゾーンが有効にも関わらず署名が無効になる事態が発生する可能性がある
 - ⇒ SOAのExpireは署名期間より短い必要がある

鍵管理

- KSK秘密鍵が漏洩すると問題
 - KSKの更新は、DSの登録変更作業が必要のため、対処に時間がかかる
- ZSKはKSKに比べリスクは小さい
 - 万が一漏洩した場合でも、KSKに比べ短時間で更新できる
- いずれにしても鍵管理は十分厳重に行う

BINDキャッシュサーバでの DNSSECの設定

古くて新しいDNSSEC

RFC	発行年	概要	対応BIND
2065	1997年	DNSSEC最初のRFC	
2535	1999年	RFC 2065の改良版	9.2系まで
3658	2003年	DS RRの登場	9.3系から
4033 4034 4035	2005年	現行のDNSSEC方式の基本 (NSEC 方式)	9.3系から
5011	2007年	トラストアンカーの自動更新	9.7系から
5155	2008年	NSEC3 方式(NSEC方式の改良)	9.6系から
5702	2009年	DNSKEY,RRSIGの SHA-2 対応	9.7系から

BINDキャッシュサーバでの DNSSECの設定

- バージョンは9.7.2-P3以降を使う
 - JPやrootゾーンではアルゴリズムに**RSASHA256**を採用している
- 通常のカッシュサーバの設定に、署名の検証を行う設定を追加する
 - named.conf の options 部分以下を追加する
 - dnssec-enable** **yes;**
 - dnssec-validation** **yes;**
 - 署名の検証に必要な情報を登録する
⇒ **トラストアンカーの登録**

署名の検証を行うオプション

- dnssec-enable
 - DNSSEC対応にするかどうかのオプション
 - BIND 9.4以降のデフォルト yes
- dnssec-validation
 - DNSSECの署名検証を行うかどうかのオプション
 - BIND 9.4のデフォルト no
 - BIND 9.5以降のデフォルト yes

```
options {  
    ....  
    dnssec-enable            yes;        // BIND 9.7 であれば  
    dnssec-validation       yes;        // 設定しなくてもよい  
    ....  
};
```

rootゾーンの公開鍵

- rootゾーンの公開鍵関連情報の入手先
<https://data.iana.org/root-anchors/>
- rootゾーンの公開鍵
<https://data.iana.org/root-anchors/root-anchors.xml>
 - ただし**DS情報のみ**
 - BINDで設定するトラストアンカーはKSK公開鍵
⇒ KSK公開鍵を入手する必要がある(後述)
 - DNSSEC Trust Anchor Publication for the Root Zone
<https://data.iana.org/root-anchors/draft-icann-dnssec-trust-anchor.html>
- 他に、PGPの公開鍵、署名等がある

root-anchors.xmlの内容

```
<?xml version="1.0" encoding="UTF-8"?>
<TrustAnchor
  id="AD42165F-3B1A-4778-8F42-D34A1D41FD93"
  source="http://data.iana.org/root-anchors/root-anchors.xml">
<Zone>.</Zone>
<KeyDigest id="Kjqmt7v" validFrom="2010-07-15T00:00:00+00:00">
  <KeyTag>19036</KeyTag>
  <Algorithm>8</Algorithm>
  <DigestType>2</DigestType>
  <Digest>
    49AAC11D7B6F6446702E54A1607371607A1A41855200FD2CE1CDDE32F24E8FB5
  </Digest>
</KeyDigest>
</TrustAnchor>
```

rootのKSK公開鍵の入手

- rootゾーンのKSK公開鍵を入手

```
$ dig . dnskey | grep -w 257 > root-ksk.key
```

– 257は現在有効なKSK (ZSKは256)

- KSK公開鍵からDSを生成

```
$ dnssec-dsfromkey -2 -f root-ksk.key .  
. IN DS 19036 8 2 49AAC11D<中略>F24E8FB5
```

- 結果をroot-anchors.xmlと比較し、差異の無いことを確認する

注意: httpsを信頼の基点として作業
PGP鍵で検証するのが望ましい

BINDへトラストアンカーの登録

- named.conf にトラストアンカーを登録
 - root-ksk.keyから “<TTL> IN DNSKEY” を除いて trusted-keysに設定
- trusted-keysの書式
 - ドメイン名 数字 数字 数字 公開鍵
 - 公開鍵は “ “ で囲み、空白、TAB、改行等があってもよい
 - 複数ドメイン名の設定が可能

```
trusted-keys {  
    "."      257 3 8  
            "AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVL0yQ  
            bSEW008gcCjFFVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh  
            <中略>  
            LmqrAmRLKBP1dfwhYB4N7knNnulqQxA+Uk1ihz0==" ;  
};
```

キャッシュサーバの動作確認

- named.conf の変更が終わったら、キャッシュサーバ用のnamedを再起動する
- digコマンドでDNSSEC対応ゾーンの確認
 - “.” の SOAが確実
 - 代表的なDNSSEC対応のドメイン名
www.pir.org, www.isc.org, www.iana.org
www.iis.se 等

キャッシュサーバへのdigの結果

```
$ dig @127.0.0.1 +dnssec www.isc.org a

; <<>> DiG 9.7.1-P2 <<>> @127.0.0.1 +dnssec www.isc.org a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7369
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 13

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

;; ANSWER SECTION:
www.isc.org.                455        IN          A           149.20.64.42
www.isc.org.                455        IN          RRSIG      A 5 3 600 20101021233649 20100921233649 31518 isc.org. gh5y8VWKqWdVgPkK<略>

;; AUTHORITY SECTION:
isc.org.                    43054     IN          NS          ns.isc.afilias-nst.info.
isc.org.                    43054     IN          NS          sfba.sns-pb.isc.org.
isc.org.                    43054     IN          NS          ord.sns-pb.isc.org.
isc.org.                    43054     IN          NS          ams.sns-pb.isc.org.
isc.org.                    43054     IN          RRSIG      NS 5 2 43200 20101021233649 20100921233649 31518 isc.org. G+qmwMI9fPrMV <略>
<以下略>
```

digの結果のflagsフィールド

- flags: qr rd ra **ad**;
 - DNSにおけるさまざまな状態を表すフラグ
- DNSSECに関するflag
 - ad: Authentic Data
署名が検証できた正しいデータであることを示す
 - cd: Checking Disabled
署名のチェックを行っていない状態を示す
- 署名の検証を行わない場合は+cdを指定

```
$ dig @127.0.0.1 +cd www.isc.org a  
flags: qr rd ra cd;
```

- BINDはtrusted-keysを設定すると内部では**必ず署名の検証を行う**

DNSSEC検証の失敗

- トラストアンカーの設定を誤った場合

```
$ dig +dnssec . soa  
status: SERVFAIL
```

⇒ 答えが得られない

- 署名検証に**失敗した**場合、名前解決**不能**
⇒ DNSSEC最大の**リスク**

トラストアンカーの自動更新

- rootのKSKが更新された場合、BIND等で設定しているトラストアンカーの更新が必要
 - 定期的な更新作業を要求される
- トラストアンカーの自動更新機能
 - RFC 5011対応 - BIND 9.7の新機能の一つ
- rootのKSK管理は、RFC 5011に準拠
 - RFC 5011対応の設定を行えば、トラストアンカーの更新作業を自動化できる

RFC 5011対応の設定(1/2)

- managed-keysにトラストアンカーを設定
 - trusted-keysの代わりにmanaged-keysを使う
 - 別ドメイン名であれば併用も可能
- managed-keysの書式
 - ドメイン名 initial-key 数字 数字 数字 公開鍵
 - 「initial-key」を追加しあとはtrusted-keysと同じ
 - 複数ドメイン名の設定が可能

```
managed-keys {  
    "." initial-key 257 3 8  
    "AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVL0yQ  
    bSEW008gcCjFFVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh  
    <中略>  
    LmqrAmRLKBP1dfwhYB4N7knNnulqQxA+Uk1ihz0==" ;  
};
```

RFC 5011対応の設定(2/2)

- managed-keysでは、2つのファイルがワーキングディレクトリに作成される
 - managed-keys.bind KSKの履歴
 - managed-keys.bind.jnl 上記のジャーナルファイル
 - ディレクトリはmanaged-keys-directoryで変更可
- ディレクトリのパーミッションを、namedの実行権限で読み書きできるように設定する

```
$ ps auxc | fgrep named
bind          932  0.0  0.8 48576 31844  ?? Ss 29Jul10 2:14.61 named
$ ls -ld /var/run/named
drwxr-xr-x 2 bind wheel 512 Jul 30 12:24 /var/run/named
```

- 運用事例が少ないため、RFC 5011の**過信は禁物**

鍵生成と署名作業

DNSSEC鍵の作成: dnssec-keygen

- -a 鍵生成アルゴリズムの指定
 - RSASHA1、NSEC3RSASHA1、RSASHA256、RSASHA512などを指定する
- -b ビット長
 - ZSK: 上記アルゴリズムの場合 1024ビット以上
 - KSK: 上記アルゴリズムの場合 2048ビット以上
- -f ksk
 - KSKを作成する場合に指定
- 最後に名前(ゾーン名)を指定

KSKとZSKの生成

- ZSKの生成

```
# dnssec-keygen -a RSASHA256 -b 1024  
example.jp > zsk-example.jp
```

– 鍵のファイル名を表示するので、その結果を保存する
Kexample.jp.+008+52863

3桁の数字はアルゴリズム、5桁は識別子(ID)

- 1組の鍵ファイルができる

Kexample.jp.+008+52863.key ⇒ 公開鍵

Kexample.jp.+008+52863.private ⇒ 秘密鍵

- KSKの生成

```
# dnssec-keygen -a RSASHA256 -b 2048 -f ksk  
example.jp > ksk-example.jp
```

鍵ファイルの中身の例(秘密鍵)

```
Private-key-format: v1.3
Algorithm: 8 (RSASHA256)
Modulus:
  yRDBKqRGEZUj6gAB2rd4SznfcHb9sQMU9rq/BHhTs7FHxt90ey7cG2rLrh28l4AY4tFWGSKBN4bPQhVWxvT69zqu4jBuy7Gg7j2Rs
  +Eb+WYGUAyUWQM9MvPGOTD0tImE4m5kjUTnoziS/GFDSMj8GXs4rHmlrUWyClf9lv2Ru0=
PublicExponent: AQAB
PrivateExponent:
  rhnf6biNI7RsgLa45FZxx0wYnB2s1pXAlVRnCsVWToZ3jHD5P6D33pW/AGmnX9f/tIdnciQ6l4YX+TTYsSiYFemvG4TRZcx91rY5J
  VVxBL9lkHPqTiNwdrdM5mcKE3835SFK/XEInPYHErAGi2Gz3bQG1k2dPGjQG6ai5ualx9E=
Prime1: 9Gy+ms2q0EYpHVb4drHR25l4HzC2xpdgH77/0pwtftZSg7NZixb2YI9ULRy38y+57YtZ2AZ4s51QzgGHpt2xUw==
Prime2: 0pZZH5hPkkjNFVTBKQc0nve3Pd/FnS7GmlOhq2NhXQEYexRkYt0R2VYfV+GYgszuooOXqjRWi0eI1G/uMfPevw==
Exponent1: lxLboJz8Ne0Xnn3R5rMzzbKGz2hxoD+R9y07u7YyXJilwCdLci/IKpiMY7G7dMEL/2nBJ0egtQvIFPxW1qF55w==
Exponent2: CxRN7BofXBrob07eOsJeSl7ODTtQskxbtpLf1pyL6tC78P3JqknnPoABdiYwV/FgPLyfpzhK0NkaodKhvY8PEQ==
Coefficient: 8q2G3F5fagdIvejnm6Jt7kXD5EYI3LXOVGwDYgZOLH6vPF/Eh5952Q9ivSr7qNqyJWzqMEPlfpET4uhRLizy5Q==
Created: 20101124065848
Publish: 20101124065848
Activate: 20101124065848
```

- BIND 9.7系のdnssec-keygenで作成
 - 9.6系まではformatがv1.2で、v1.2はBIND 9.7系のツールでも扱えるが、v1.3は9.7系(以降)のツールのみ対応
 - RSASHA256やRSASHA512はBIND 9.6.2以降で対応

鍵ファイルの中身の例(公開鍵)

```
; This is a zone-signing key, keyid 52863, for example.jp.  
; Created: 20101124065848 (Wed Nov 24 15:58:48 2010)  
; Publish: 20101124065848 (Wed Nov 24 15:58:48 2010)  
; Activate: 20101124065848 (Wed Nov 24 15:58:48 2010)  
example.jp. IN DNSKEY 256 3 8  
AwEAAckQwSqkRhGVI+oAAAdq3eEszX3B2/bEDFPa6vwR4U7OxR8bfdHsu  
3Btqy64dvJeAGOLRVhkigTeGz0IVVl70+vc6ruIwbsuxoO49kbPhG/lm  
BlAMlFkDEPTLzxjkw9LSJh0JuZI1E56M4kvxhQ0jI/Bl7OKx5ta1Fsgp  
X/Zb9kbt
```

- BIND 9.7系のdnssec-keygenで作成
– “;”のコメント部については後述

dnssec-keygenの注意点

- KSKとZSKの区別に注意する
 - 2組の鍵ファイル(計4個)ができ、見た目での識別は困難
- 実行時に鍵ファイル名を保存すると良い

```
# dnssec-keygen -f ksk .. > ksk-...  
# dnssec-keygen ..... > zsk-...
```


ゾーンへの署名 : dnssec-signzone

- 署名対象ゾーンファイル、ZSK、KSKを準備
 - 同じディレクトリに用意し、ゾーンファイルはゾーン名とファイル名を一致させると便利

example.jp

Kexample.jp.+008+56449.key KSK公開鍵

Kexample.jp.+008+56449.private KSK秘密鍵

Kexample.jp.+008+52863.key ZSK公開鍵

Kexample.jp.+008+52863.private ZSK秘密鍵

ゾーンへの署名(続き)

- ゾーンファイルにKSK、ZSKの公開鍵を登録
 - 公開鍵をまとめたファイルを用意し、\$INCLUDE文を利用してゾーンファイルから参照する

```
# cat `cat ksk-example.jp`.key  
    `cat zsk-example.jp`.key > example.jp.keys
```

```
; ゾーンファイル中でkeyファイルを参照  
$INCLUDE example.jp.keys
```

- SOAシリアル値の管理は、dnssec-signzoneの -N オプションにまかせるのがベター

署名前のゾーンファイル example.jp

```
$TTL      1D
$INCLUDE example.jp.keys
@         IN          SOA      ns root (
                        1          ; Serial
                        10800     ; Refresh
                        3600      ; Retry
                        3600000   ; Expire
                        1800     ) ; Minimum TTL

                        NS       ns
                        MX       10 mail

;
ns        A          192.0.2.17
www       A          192.0.2.18
mail      A          192.0.2.19

sub1      NS         ns.sub1
ns.sub1   A          192.0.2.49

sec3      NS         ns.sec3
ns.sec3   A          192.0.2.65
$INCLUDE ../sec3.example.jp/dsset-sec3.example.jp.

sub3      NS         ns.sub3
ns.sub3   A          192.0.2.81
```

署名の実行

- `dnssec-signzone -H <繰り返し回数> -3 <salt>`
 `-N <SOAのシリアル値> -k <KSK>`
 `<ゾーンファイル> <ZSK>`

```
# dnssec-signzone -H 3 -3 123ABC -N unixtime  
    -k `cat ksk-example.jp`  
    example.jp `cat zsk-example.jp`
```

- -3はNSEC3方式を選びソルトを指定するオプション
- 秘密鍵を明示的に指定する必要は無い
- 出力ファイル
 - `example.jp.signed` 署名済みのゾーン
 - `dsset-example.jp.` ゾーンへのDS RR

署名済みのゾーンファイル(抜粋)

```
; File written on Wed Nov 24 16:07:14 2010
; dnssec_signzone version 9.7.2-P2
example.jp.          86400   IN  SOA  ns.example.jp. root.example.jp. (
                    1290582434 ; serial
                    <中略>
                    )
                    86400   RRSIG SOA 8 2 86400 20101224060714 (
                    20101124060714 52863 example.jp.
                    WcEe7OoXanQAPuS8pTwYJ1wLRFkC75/2kwii
                    <中略>
                    8NmbrA3wlhoCqwEBAeQX+ZhKZtQ= )
                    86400   NS      ns.example.jp.
                    86400   RRSIG NS 8 2 86400 20101224060714 (
                    20101124060714 52863 example.jp.
                    bLcTvbBVftz6NBTSzvj8Yn0G0PyMbTYfzEvd
                    <中略>
                    Q2kG0oVtpxJoG69h0od036w+Yj8= )
                    86400   MX      10 mail.example.jp.
                    86400   RRSIG MX 8 2 86400 20101224060714 (
                    20101124060714 52863 example.jp.
                    OiKiNz7CGAnBXdgfaUm23+/VOGaLfgMk6RYB
                    <中略>
                    0D3RCXYCUoGtKdCswIM77w0U2GE= )
                    86400   DNSKEY  256 3 8 (
                    AwEAAckQwSqkRhGVI+oAAdq3eEszX3B2/bED
                    <中略>
                    4kvxhQ0jI/B17OKx5ta1FsgpX/Zb9kbt
                    ) ; key id = 52863
                    86400   DNSKEY  257 3 8 (
                    AwEAAauHCuMQzCBUaaQLNf/FPRGqcPupOdYU
                    <中略>
                    WnegM3YXJyvpSS0gZ9ykoolReqa/94XL9HCl
                    vIZUhkdczjyo/0EKdgBt5IU=
                    ) ; key id = 56449
```

DSの登録

- dsset-example.jpの内容を親ドメインに登録
 - 子ゾーンの署名時に生成されたもの
 - 内容の例

```
example.jp. IN DS 56449 8 1 6EFE<中略>4BE  
example.jp. IN DS 56449 8 2 7D29<中略>F32852 DE98ED8C
```

- DSレコードはKSKの公開鍵からも生成可能
 - dnssec-dsfromkeyコマンドを使用する

BIND権威サーバでの DNSSECの設定

BINDの設定: 権威サーバ(1/2)

- DNSSECを有効にする
 - named.conf の options 部分に dnssec-enable yes; を追加

```
options {  
    <省略>  
    dnssec-enable yes; // BIND 9.4以降は  
                        // デフォルトが yes;  
    <省略>  
};
```


BINDの設定: 権威サーバ(2/2)

- ゾーンファイルを署名済みのものに変更

```
zone "example.jp" {  
    type master ;  
    // file "example.jp.zone" ;  
    file "example.jp.signed" ;  
} ;
```

- namedを再起動

```
rndc reload
```

権威サーバの動作確認

- digコマンドでDNSSEC対応ゾーンの確認

```
# dig +dnssec +norec @127.0.0.1 www.example.jp a
```

+dnssec DNSSECを有効にする問合せ

このオプションなしでは通常の(DNSSECでない)ものと同じ結果が返る

+norec 非再帰問合せ

キャッシュサーバから権威サーバへの問合せと同じ形式の問合せ

権威サーバへのdigの結果(1/2)

```
$ dig @127.0.0.1 +noredc www.example.jp a
```

```
; <<>> DiG 9.7.2-P2 <<>> @127.0.0.1 +noredc www.example.jp
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59143
;; flags: qr aa ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;www.example.jp.                IN      A

;; ANSWER SECTION:
www.example.jp.                86400   IN      A      192.0.2.18

;; AUTHORITY SECTION:
example.jp.                    86400   IN      NS     ns.example.jp.

;; ADDITIONAL SECTION:
ns.example.jp.                 86400   IN      A      192.0.2.17

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Nov 24 16:23:02 2010
;; MSG SIZE rcvd: 81
```

+dnssec無しの
digの応答

権威サーバへのdigの結果(2/2)

```
$ dig @127.0.0.1 +dnssec +nored www.example.jp
```

```
; <<> DiG 9.7.2-P2 <<> @127.0.0.1 +dnssec +nored www.example.jp
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 21018
;; flags: qr aa ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 3

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

;; ANSWER SECTION:
www.example.jp.                86400   IN      A      192.0.2.18
www.example.jp.                86400   IN      RRSIG  A 8 3 86400 20101224060714 20101124060714 52863 example.jp.
                               r8cI/2zMc5fCtlie6pjZoaI0o0myBWzir3ykFpDpkMGWxaFlpHcCCzK ogZXfDvWlPMKP3Hc+TshZThW+FuCYlEnCsPz15n1Rv139tsP83ZHFGZ6
                               PKH7FsxGZUFtwa0cyILkckRF7BPvrITck+y0oWivzDr1LHGR+5F6hx1p 4ac=

;; AUTHORITY SECTION:
example.jp.                    86400   IN      NS      ns.example.jp.
example.jp.                    86400   IN      RRSIG  NS 8 2 86400 20101224060714 20101124060714 52863 example.jp.
                               bLcTvbBVftz6NBTSzvJ8Yn0G0PyMbTYfzEvdwpL0WYmd6zbDiX3ZGXsv EymEaWi8CDzmnBQqz5VM5dCAe5IaddZqHgAdeJJ2MRZCu2OxDdCjwEne
                               pNnFnu0TXCVyP6Dipq61mcc+2lZHLakzQ2kGOoVtpxJoG69hOod036w+ Yj8=

;; ADDITIONAL SECTION:
ns.example.jp.                 86400   IN      A      192.0.2.17
ns.example.jp.                 86400   IN      RRSIG  A 8 3 86400 20101224060714 20101124060714 52863 example.jp.
                               vGOD3t5bklnTeoZwDeXjKcZAeXFblB+qfmZzz3P7+JUFA7/1NjQGvWwO dqtGlBznFuTl+7lediOJnf9zDaJjJI7dobv10Nb3Wy/1QmZHw3hAmbcm
                               64DgbN/004j1HUORp30UgB59/Esb8HFARQQcSkRAXa7iq1gdTm5dH5oa PBO=

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Nov 24 16:28:36 2010
;; MSG SIZE rcvd: 602
```

+dnssecありでは、RRSIG RR
を加えたものが返る

DO(DNSSEC OK)ビット

- dig の +dnssec オプション
 - 問合せでEDNS0を使い、DOビットをONにすると共に512バイトを超えるサイズのDNSパケットを受けられることを宣言する
 - DNSSECでは**EDNS0のサポートは必須**
- DOビット
 - DNSSEC OK ⇒ DNSSECの応答を受ける ⇒ DNSSECを要求する
 - 権威サーバは、問合せのDOビットがONであれば、DNSSECの情報を含んだ応答を返す

スマート署名 (Smart signing) BIND 9.7での新機能

DNSSEC for Humans

- BIND 9.7から導入された、DNSSECの設定をより簡単に行う一連の機能
 - スマート署名(Smart signing)
 - 全自動ゾーン署名(後述)
 - RFC 5011への対応
 - Dynamic Update設定の簡素化
 - DLVの自動設定
- スマート署名
 - KSKやZSKの鍵管理の自動化

スマート署名

example.jpをNSEC3方式で署名

```
# ls
example.jp
# dnssec-keygen -3 example.jp
Generating key pair.....+++++ .....+++++
Kexample.jp.+007+31760
# dnssec-keygen -3 -f ksk example.jp
Generating key pair.....+++ .....+++
Kexample.jp.+007+22740
# dnssec-signzone -3 aabbcc -S example.jp
Fetching ZSK 31760/NSEC3RSASHA1 from key repository.
Fetching KSK 22740/NSEC3RSASHA1 from key repository.
Verifying the zone using the following algorithms: NSEC3RSASHA1.
Zone signing complete:
Algorithm: NSEC3RSASHA1: KSKs: 1 active, 0 stand-by, 0 revoked
                        ZSKs: 1 active, 0 stand-by, 0 revoked

example.jp.signed
# ls
Kexample.jp.+007+22740.key          dsset-example.jp.
Kexample.jp.+007+22740.private    example.jp
Kexample.jp.+007+31760.key        example.jp.signed
Kexample.jp.+007+31760.private
```


スマート署名

- dnssec-signzoneの-Sオプション
 - 鍵情報を自動的に取り込むため、ゾーンファイルは通常のゾーンファイル(非DNSSEC)のままよい
- dnssec-keygenの便利なデフォルト値
 - アルゴリズム RSASHA1
 - 「-3」を指定した場合 NSEC3RSASHA1
 - ZSKのbit長 1024 bit
 - KSKのbit長 2048 bit
- -K 鍵を保存するディレクトリ(Key repository)を指定するオプション
 - 他の鍵関連コマンドでも指定できる

日付情報を指定するオプション

- dnssec-keygenで鍵生成時に指定
 - P ゾーンへの出力時刻(Publication date)
 - R 鍵の破棄時刻(Revocation date)
 - A 署名鍵としての使用開始時刻(Activation date)
 - I 署名鍵使用終了時刻(Inactivation date)
 - D ゾーンからの削除時刻(Deletion date)
 - -Pと-Aのデフォルトは now (現在時刻)
 - 絶対時刻: YYYYMMDD 又は YYYYMMDDHHMMSS
 - 相対時刻: +数字 又は -数字 ‘y’, ‘mo’, ‘w’, ‘d’, ‘h’, or ‘mi’ (年、月、週、日、時間、分)を指定可能
- dnssec-settime - 日付情報を変更するコマンド

日付を指定して鍵を作成(1/2)

```
# mkdir keys ①
# dnssec-keygen -K keys -f ksk example.jp ②
Kexample.jp.+005+45154
# dnssec-keygen -K keys -P now -A now -D +31d example.jp ③
Kexample.jp.+005+20076
# dnssec-keygen -K keys -P now -A +30d -D +61d example.jp ④
Kexample.jp.+005+45870
# dnssec-signzone -K keys -N unixtime -S example.jp ⑤
Fetching KSK 45154/RSASHA1 from key repository.
Fetching ZSK 20076/RSASHA1 from key repository.
Fetching ZSK 45870/RSASHA1 from key repository.
Verifying the zone using the following algorithms: RSASHA1.
Zone signing complete:
Algorithm: RSASHA1: KSKs: ① active, 0 stand-by, 0 revoked
                    ZSKs: ① active, ① stand-by, 0 revoked
example.jp.signed
# ls -F ⑥
dsset-example.jp.          example.jp.signed
example.jp                 keys/
```

日付を指定して鍵を作成(2/2)

- ① 鍵用のディレクトリ(keys)を作成
- ② KSKを作成
- ③ 最初に使うZSKを作成
すぐに署名に使用し、31日後にゾーンから削除
- ④ 2番目に使うZSKを作成
すぐにゾーンに出力、30日後に署名に使用
- ⑤ ゾーンへの署名
KSKは1個、ZSKは1個が署名用1個が事前公開用
この例ではNSEC方式を採用し、SOAのシリアルは
unixtimeを使っている
- ⑥ 鍵はkeysディレクトリ内にある

署名と鍵更新の自動化

- cron等を利用する
 - 定期的に dnssec-keygen で -P, -A, -I, -D を適正に設定したZSKを作成
 - 定期的に dnssec-signzone -S で再署名
- ⇒ 鍵更新、再署名の自動化が可能になる
- KSKについても同様の処理が可能
 - 但しDSの更新には親ゾーンとのやり取りが必要なため、完全な自動化は難しい

注意:

dnssec-keygenで-Aを指定する場合、必ず-Pも指定する
⇒ BIND 9.7.2-P3時点での不具合

全自動ゾーン署名 BIND 9.7での新機能

全自動ゾーン署名

- namedがゾーンへの署名、鍵更新を行う
 - dnssec-signzoneは利用しない
 - 鍵にはスマート署名の場合と同様、日付情報を設定する
- ゾーン毎に次のいずれかを設定する
 - auto-dnssec allow; 署名等はrndcコマンド使って別途制御する(定期的な再署名は行われる)
 - auto-dnssec maintain; 鍵ファイルに記録されている日付情報に基づいて完全に自動化する

全自動ゾーン署名設定の例

```
options {
    ....
    directory          "/var/named";
    session-keyfile  "/var/named/session.key";
    ....
};

zone {
    type              master;
    file              "master/example.jp";
    key-directory    "master/keys";
    update-policy   local;
    auto-dnssec     maintain;
};
```


全自動ゾーン署名設定(1/2)

- session-keyfile
 - ダイナミックアップデートのための鍵ファイルの指定
 - 指定しない場合コンパイル時のデフォルトが適用される
- key-directory
 - スマート署名の-Kで指定するディレクトリと同じもの
- update-policy
 - ダイナミックアップデートのポリシーの設定で、ここでは単純な local を設定
 - 必要に応じて他のポリシーも設定できる

全自動ゾーン署名設定(2/2)

- ゾーンファイルは、非DNSSECのものでよい
- rndcコマンドが正しく動作するよう設定する
- ゾーンファイル、ゾーンファイルのあるディレクトリなどは、namedプロセスの権限で書き換え可能なパーミッションに設定
 - 必要に応じてnamedがファイルを作成したり、書き換えたりするため

auto-dnssec maintain;

- named 起動後、鍵ディレクトリ内の鍵ファイルの日付に応じてゾーンに署名を行う
 - dnssec-signzone -Sと同様の動作となる
- KSKとZSKをNSEC3用に設定しても、デフォルトはNSECとなる(DNSSECとしては問題無い)
- NSEC3で運用するための二つの方法
 - ダイナミックアップデート(nsupdateコマンド)を使い、NSEC3PARAMレコードを追加する
 - ⇒ 追加直後にNSEC3方式に切り替わる
 - 予めゾーンファイルにNSEC3PARAMレコードを登録
 - ⇒ 起動直後の最初の署名でNSEC3方式になる

nsupdateコマンド

- 稼働中のゾーンデータに、動的にRRの追加削除(ダイナミックアップデート)を行うコマンド
- NSEC3PARAM RRを追加する例

```
# nsupdate -k /var/named/session.key -l
> update add example.jp 0 nsec3param 1 0 5 AABBCDD
> send
> quit
#
```

- -l (エル)でローカルのnamedを指定
- 更新情報はジャーナルファイルに記録される
 - この例では"/var/named/master/example.jp.jnl"になる

全自動ゾーン署名時の ゾーン情報の変更

- ゾーンファイルはnamedが直接管理するため、単純には編集できない
- 解1: ダイナミックアップデートを利用する
 - nsupdateコマンドを利用して、RRの追加、削除、変更を行う
- 解2: 一時的にnamedがゾーンファイルを更新するのを停止させ通常通り編集し、namedのゾーンファイルの更新を再開する

全自動ゾーン署名の ゾーンファイルの編集

- 一時的にゾーンファイルの更新を停止する

```
# rndc freeze example.jp
```

- この時点でnamedが保持しているゾーン情報がすべてexample.jpのゾーンファイルに反映される

- 編集する

```
# vi example.jp
```

- RRSIGなどが追加されているが気にしなくて良い

- ゾーンファイルの更新を再開する

```
# rndc thaw example.jp
```

- 再開した時点でnamedがゾーンデータを読み込み、再署名が行われる

鍵の追加と署名

- ZSKの追加(KSKも同様)
 - dnssec-keygenで日付情報を適正に指定したZSKを生成し鍵のディレクトリに用意する
 - 鍵ファイルの追加後rndcコマンドで鍵情報の変更をnamedに通知

```
# rndc loadkeys example.jp
```

- dnssec-settimeで鍵の日付情報を変更した場合も同様の処理が必要
- DNSSEC運用に必須の定期的な再署名は自動的に行われる
 - なんらかの理由により署名を行いたい場合

```
# rndc sign example.jp
```

全自動ゾーン署名の注意点

- DS RRの作成
 - dnssec-signzoneを利用しないため、DS RRは、KSKの鍵ファイルから作成する必要がある

```
# dnssec-dsfromkey Kexample.jp.+005+35251.key >  
dsset-example.jp.
```

- 長期間運用を続けると鍵ファイルが増える
 - 使い終わった鍵ファイルを削除する仕組みを、別途用意する必要がある
 - ⇒ スマート署名の場合も同じ

運用面から見た全自動ゾーン署名

- 比較的運用が単純化できる
- ダイナミックアップデートを使う場合は差分情報のみ署名されるため、ゾーン情報変更時の負荷が軽い
- ゾーンファイル内にRRSIGやNSEC等が自動的に追加されるため、ゾーンファイルの更新履歴を記録しにくい
- 運用実績があまり無いためトラブルシューティングに対する不安が残る

運用面から見たスマート署名

- ゾーンファイルの変更履歴はとりやすい
- 署名完了後のゾーンファイルをnamedに読み込ませるため、ある程度確実な運用ができる
- dnssec-signzoneには差分署名の機能が無い
ため、大きなゾーンファイルに対しては、
ゾーン変更時の署名の負荷が大きい
 - 但しDNSSECの運用では定期的な再署名が必要であり、再署名時は全署名となるため、スマート署名だから問題になるものではない

DNSSEC化による DNSデータの変化

DNSSEC有無による www.nic.seの検索

- DNSSEC無し

```
$ dig +norec @a.ns.se www.nic.se a / grep SIZE
;; MSG SIZE rcvd: 157      (親のNSに問合せ)
$ dig +norec @ns.nic.se www.nic.se a / grep SIZE
;; MSG SIZE rcvd: 173      (自分自身のNSに問合せ)
```

- DNSSEC有り

```
$ dig +norec +dnssec @a.ns.se www.nic.se a / grep
SIZE
;; MSG SIZE rcvd: 414      (親のNSに問合せ)
$ dig +norec +dnssec @ns.nic.se www.nic.se a /
grep SIZE
;; MSG SIZE rcvd: 1180     (自分自身のNSに問合せ)
```

権威サーバへのdigの結果(1/2)

- +dnssec無しのdigの応答

```
; <<>> DiG 9.6.1 <<>> +norec @ns.nic.se www.nic.se a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14346
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 4

;; QUESTION SECTION:
www.nic.se.                IN                A

;; ANSWER SECTION:
www.nic.se.  60               IN                A                212.247.7.218

;; AUTHORITY SECTION:
nic.se.      3600            IN                NS                ns3.nic.se.
nic.se.      3600            IN                NS                ns2.nic.se.
nic.se.      3600            IN                NS                ns.nic.se.

;; ADDITIONAL SECTION:
ns.nic.se.   3600            IN                A                212.247.7.228
ns.nic.se.   3600            IN                AAAA             2a00:801:f0:53::53
ns2.nic.se.  3600            IN                A                194.17.45.54
ns3.nic.se.  60              IN                A                212.247.3.83

;; Query time: 328 msec
;; SERVER: 212.247.7.228#53(212.247.7.228)
;; WHEN: Tue Jul 7 23:39:18 2009
;; MSG SIZE rcvd: 173
```

権威サーバへのdigの結果(2/2)

- +dnssec有り 各RRにRRSIG RRを加えたものが返る

```
; <<>> DiG 9.6.1 <<>> +norec +dnssec @ns.nic.se www.nic.se a
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5979
;; flags: qr aa; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 9

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

;; ANSWER SECTION:
www.nic.se.    60                IN                A                212.247.7.218
www.nic.se.    60                IN                RRSIG            A 5 3 60 20090714132001 20090704132001 58670 nic.se. izdsOhTB1XThccw2Wv4TZjl

;; AUTHORITY SECTION:
nic.se.        3600             IN                NS                ns2.nic.se.
nic.se.        3600             IN                NS                ns.nic.se.
nic.se.        3600             IN                NS                ns3.nic.se.
nic.se.        3600             IN                RRSIG            NS 5 2 3600 20090714132001 20090704132001 58670 nic.se. pKDbUYXLQpPnhlU9NAZh

;; ADDITIONAL SECTION:
ns.nic.se.    3600             IN                A                212.247.7.228
ns.nic.se.    3600             IN                AAAA             2a00:801:f0:53::53
ns2.nic.se.   3600             IN                A                194.17.45.54
ns3.nic.se.   60               IN                A                212.247.3.83
ns.nic.se.    3600             IN                RRSIG            A 5 3 3600 20090714132001 20090704132001 58670 nic.se. GzLodvUOd0oB4qfhhbp8H
ns.nic.se.    3600             IN                RRSIG            AAAA 5 3 3600 20090714132001 20090704132001 58670 nic.se. 0tvno8Vz7Ihm27AZ+H
ns2.nic.se.   3600             IN                RRSIG            A 5 3 3600 20090714132001 20090704132001 58670 nic.se. UcEcYGX59H8bAVGwhfwko
ns3.nic.se.   60               IN                RRSIG            A 5 3 60 20090714132001 20090704132001 58670 nic.se. NRoFeFzAm0hoyKa2ObxjCFB

;; Query time: 382 msec
;; SERVER: 212.247.7.228#53(212.247.7.228)
;; WHEN: Tue Jul 7 23:39:24 2009
;; MSG SIZE rcvd: 1180
```

注意: RRSIGは行の途中まで、残り省略

DNSSEC有無による 存在しないドメイン名の検索

- example.jp +dnssec無し

```
$ dig +norec @a.dns.jp example.jp a | grep SIZE
;; MSG SIZE rcvd: 75
```

- example.jp +dnssec有り

```
$ dig +norec +dnssec @a.dns.jp example.jp a | grep
SIZE
;; MSG SIZE rcvd: 741
```

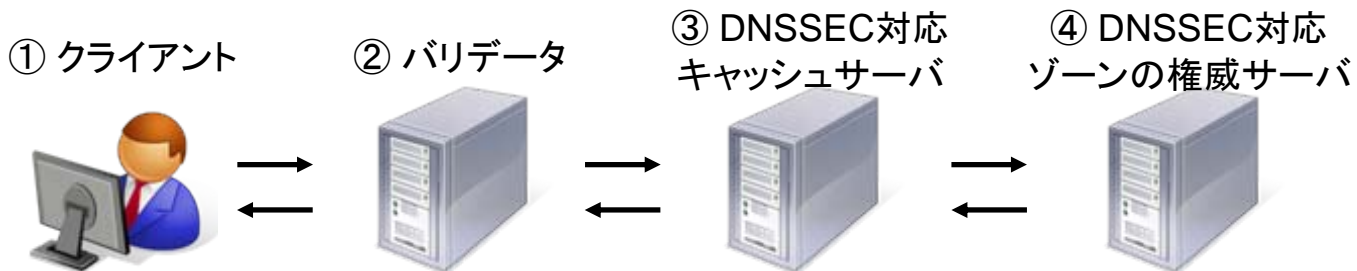
注意: example.jpは存在しないドメイン名

DNSSEC対応になると

- 署名の付加によりゾーンデータが大きくなる
 - 5～10倍程度(鍵のbit長に依存)
 - プライマリが署名すると、セカンダリにもインパクトがある
- DNS応答パケットのサイズが大きくなる
 - DNSトラフィックが増える
 - キャッシュサーバのキャッシュ効率が落ちる
 - 特に存在しない名前の検索では顕著
- DNSSEC対応のキャッシュサーバの実装では、**DNSSEC設定を行わなくてもDOビットはON**となる

何故DOビットは常時ONなのか

- キャッシュサーバ自身では署名検証を行わなくても、他の署名検証を行うもの(バリデータ)のために、署名があれば対象レコードと同時にキャッシュ



DNSSECにおいて署名検証が独立したモデル

②が署名検証を行うキャッシュサーバで③をフォワード先として指定している場合や、②は存在せず①のクライアントが直接署名検証を行う場合等

DOビットが常時ONのインパクト

- 問合せ1回あたりの応答パッケージが大きくなる
 - DNSのトラフィックが増加する
- キャッシュサーバではキャッシュに必要なメモリ量が増える
 - 場合によっては、キャッシュできるレコード数が減り、キャッシュ効率に影響する
- 手元のDNSSEC設定とは関係なく、DNSSECの普及度によって影響する

DNSSECのリスク

DNSSEC化による負荷の増加(1)

- 権威DNSサーバ側
 - 署名を作成するための負荷
⇒ 但し、DNSサーバと別サーバでの処理が可能
 - 署名が負荷によるDNSデータを保持するためのメモリ量の増加
 - 応答にRRSIGを付加するための処理
- キャッシュDNSサーバ側
 - 署名検証の負荷
但し署名検証は、キャッシュ時に行われ、キャッシュ済みであれば、改めて署名検証することは無い
 - 署名データもキャッシュするためメモリ使用量の増加

DNSSEC化による負荷の増加(2)

- キャッシュ・権威DNSサーバ共通の負荷
 - NSEC3方式の場合、ハッシュ計算コスト
- DNSSEC対応によって
 - 同じサーバであれば処理能力は3～5割程度減少する ⇒ 条件によって大きく変化する
 - メモリ使用量は5～10倍程度増加
 - ⇒ 署名検証する・しないとは独立に起きる

DNSSECの署名検証の失敗

- 署名検証に失敗した場合、名前解決不能
 - DNSSECは嘘を識別する技術
 - ⇒ 正しいものを探し出す技術ではない
- 署名検証が失敗する主な要因
 - 鍵を取り違えた
 - 上位に登録するDSを誤った
 - 署名開始前の鍵を作った時点で上位にDSに登録した
 - 署名の有効期間を過ぎた
 - etc...

サーバ時刻の同期の必要性

- 署名には有効期間があり有効期間外は無効
 - 有効期間の開始時刻、終了時刻は絶対時刻
 - 署名が正しくてもサーバの時刻が極端に違っていると、署名検証に失敗する
- DNSSEC運用を行う場合、サーバの時刻を正しく合わせる必要がある
 - NTPなどを利用するのが確実
 - 実用上は、分程度まで合っていれば問題ない

DNSSECのまとめ

従来(DNSSEC無し)との比較(1)

- ゾーン管理(権威サーバ)側
 - ZSKとKSKを作成し管理する必要がある
 - ゾーンに署名を行う必要がある
 - 定期的に鍵の更新を行う必要がある
 - 子ゾーンでは親ゾーンにDSの登録作業を行う必要がある(KSKを変更する度に必要)
- 鍵管理の手間と、ゾーン署名のコストが増えるが、ある程度は自動化可能

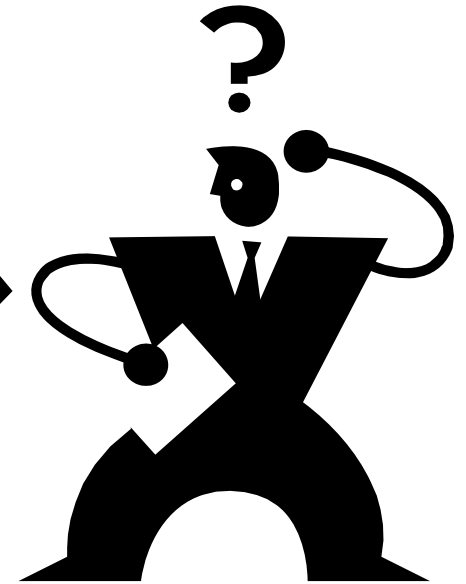
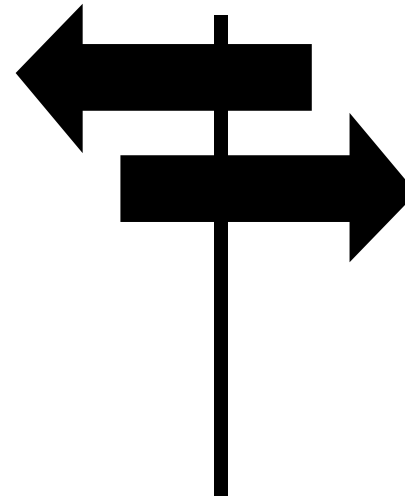
従来(DNSSEC無し)との比較(2)

- キャッシュサーバ側
 - DNSSEC機能を有効にし、トラストアンカーを設定する
 - 必要に応じてトラストアンカーを更新する
 - 署名検証による負荷の増大の懸念がある
- 双方でサーバの時刻を正しく設定する

DNSSECまとめ

- DNSSECは、公開鍵暗号技術を利用した署名によるDNSデータ保護のしくみ
 - KSKとZSKの2つの鍵を使う
 - 親ゾーンにはNSに加えてDSを登録する
 - rootゾーンのKSKの公開鍵を使って署名を検証
 - 定期的な鍵の更新と再署名とが必要

Q and A



参考 : DNSSEC 関連ツール

OpenDNSSEC

<http://www.opendnssec.org/>

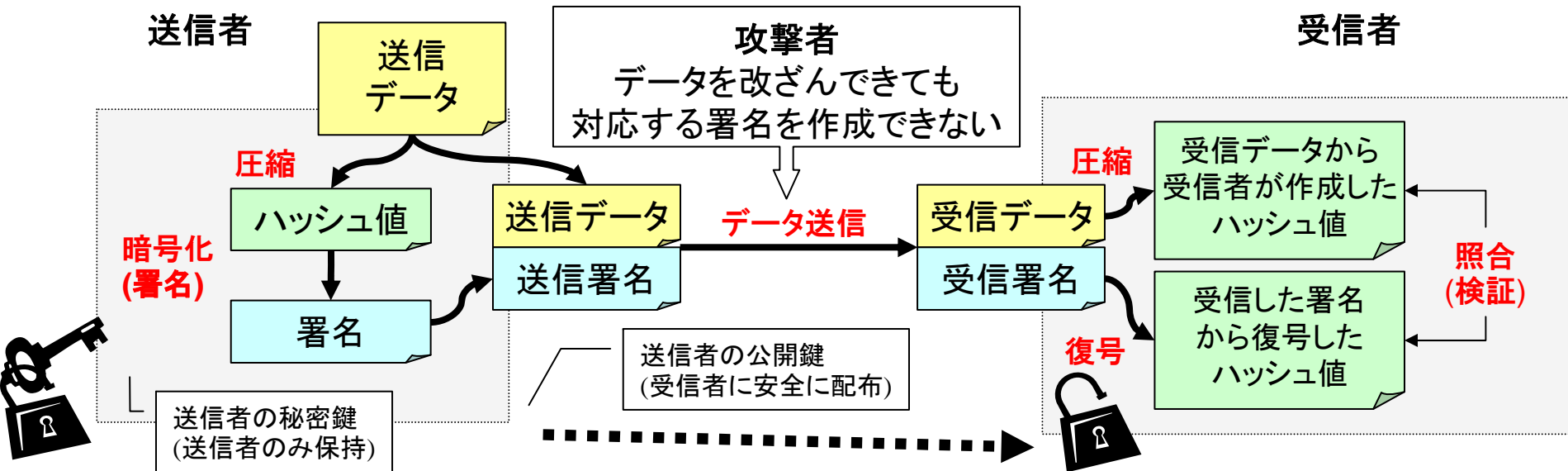
- DNSSECの運用に必要な作業の多くを自動化するツール集
 - 定期的なゾーンの再署名・鍵更新等
- TLDや、大量のDNSSEC対応ゾーンを扱う場合に適している
 - rootゾーン、SE、UK等で採用
- 多くの関連ソフトウェアを別途インストールする必要がある、インストールに手間がかかる

DNSSECで役立つWEBサイト

- DNSSEC Debugger
 - <http://dnssec-debugger.verisignlabs.com/>
 - DNSSECの署名が正しいかどうかを確認する
- A DNS visualization tool
 - <http://dnsviz.net/>
 - DNSSECの信頼の連鎖を図で表示する
- <http://test.dnssec-or-not.org/>
 - 利用中のキャッシュDNSサーバがDNSSEC対応かどうかを教えてくれるサイト

参考：電子署名とDNSSEC

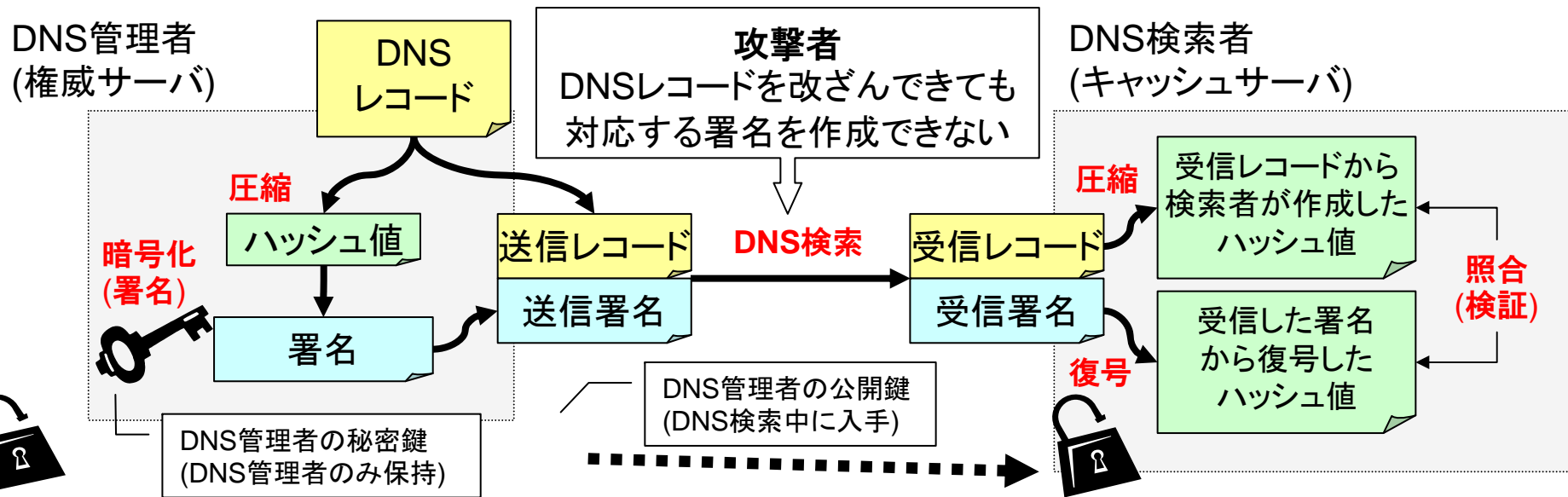
電子署名の概念



電子署名の概念

- 送信者の秘密鍵で送信データのハッシュ値を**暗号化**したものが**署名**
- 公開鍵で署名を復号すれば送信者作成のハッシュ値が得られる
- 受信データから受信者が作成したハッシュ値と、公開鍵で復号したハッシュ値が同じであるか照合(**検証**)する
- 同じであれば送信者からの完全なデータであると判断できる
 - 署名を作成できるのは送信者しかいない(出自の保証)
 - データが改ざんされていれば比較が一致しない(完全性の保証)

電子署名のDNSへの応用 (DNSSEC)



電子署名のDNSへの応用 (DNSSEC)

- DNS管理者は、**署名鍵**(秘密鍵＋公開鍵)を作成
- DNS管理者は、**DNSレコード(ハッシュ値)を秘密鍵で署名**
- 検索を受けたDNSサーバは、**DNSレコードに署名を添付**して応答
- DNS検索者は、DNS管理者の公開鍵を用いて署名を復号し、検索で得たDNSレコードと照合することで、**出自・完全性を検証**
- この仕組みを基本単位とし、DNS階層で**信頼の連鎖**を作ることによって実現