DNSSECの概要

2015年11月19日
Internet Week 2015 今日から始めるDNSSECバリデーション

株式会社日本レジストリサービス（JPRS）

原田　めぐみ

本チュートリアルの内容

- DNSSECのしくみ
- DNSSEC対応が必要な関係者
- DNSSECの鍵と信頼の連鎖
- DNSSECバリデーションの概要
- まとめ

- 参考資料：DNSSECのリソースレコード(RR)

DNSSECのしくみ

DNSキャッシュポイズニング攻撃とその対策

- DNSキャッシュポイズニング攻撃とは？
  - DNSに対する攻撃手法の一つ
  - 正しい権威DNSサーバーからの応答がフルリゾルバー(キャッシュDNSサーバー)に届く前に、偽装したDNS応答パケットを送り込む
- 従来のDNSキャッシュポイズニング攻撃の対策
  - フルリゾルバーに対するアクセスコントロールの実施
    - 外部からの問い合わせを制限する
  - ソースポートランダマイゼーションの適用
    - 問い合わせポートをランダム化して、攻撃の成功確率を下げる

⇒上記は攻撃の成功率を低減することしか出来ない

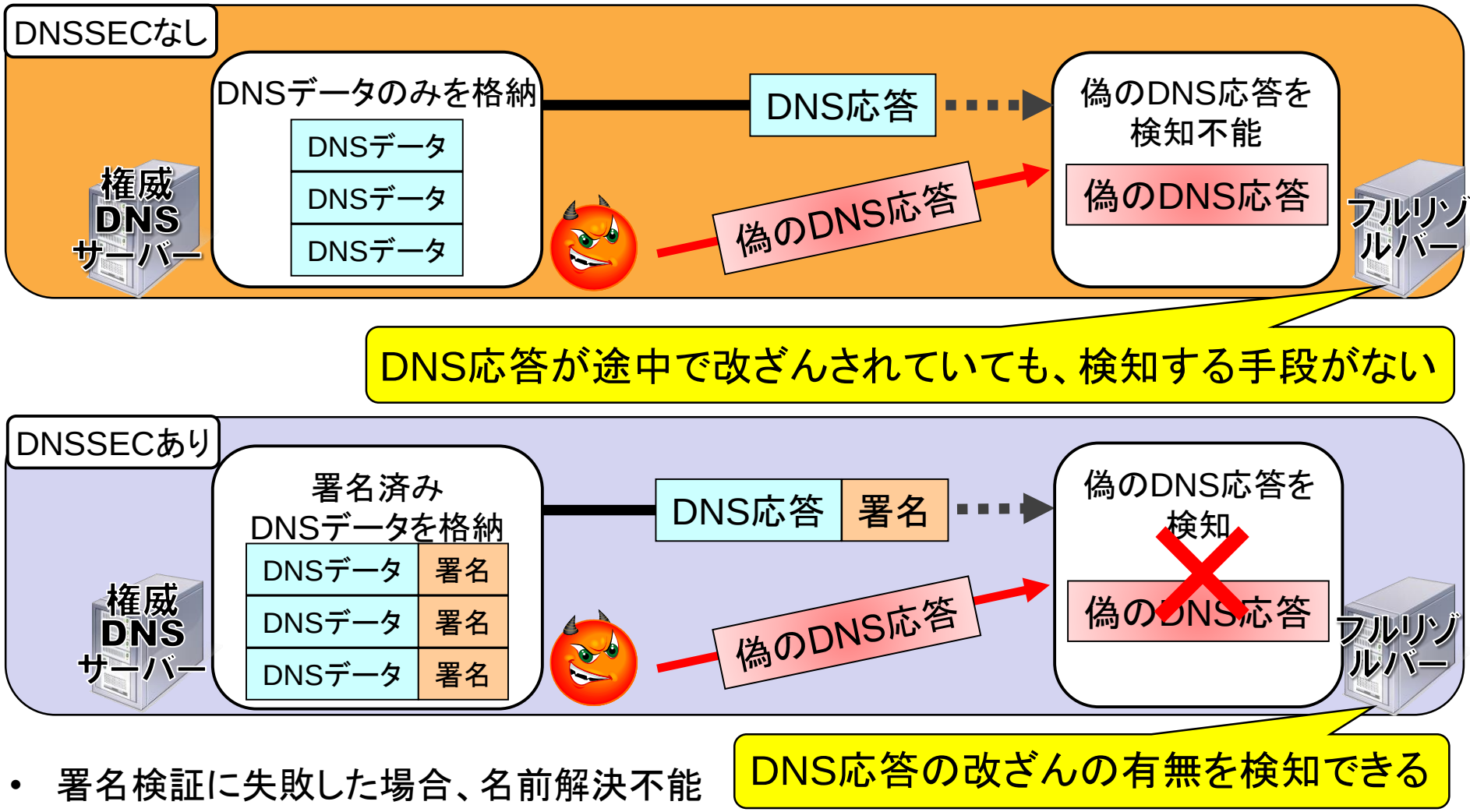
⇒根本的な対策として、DNSSECが標準化された

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
- DNSSECの対象範囲
  - 対象としているもの
    - 出自の保証
      - DNS問合せの回答が、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
    - 完全性の保証
      - DNS問合せの回答における、DNSレコードの改変の検出
  - 対象としていないもの
    - DNS問合せ/回答内容の暗号化

    ※ DNSレコードは公開情報という考え方から

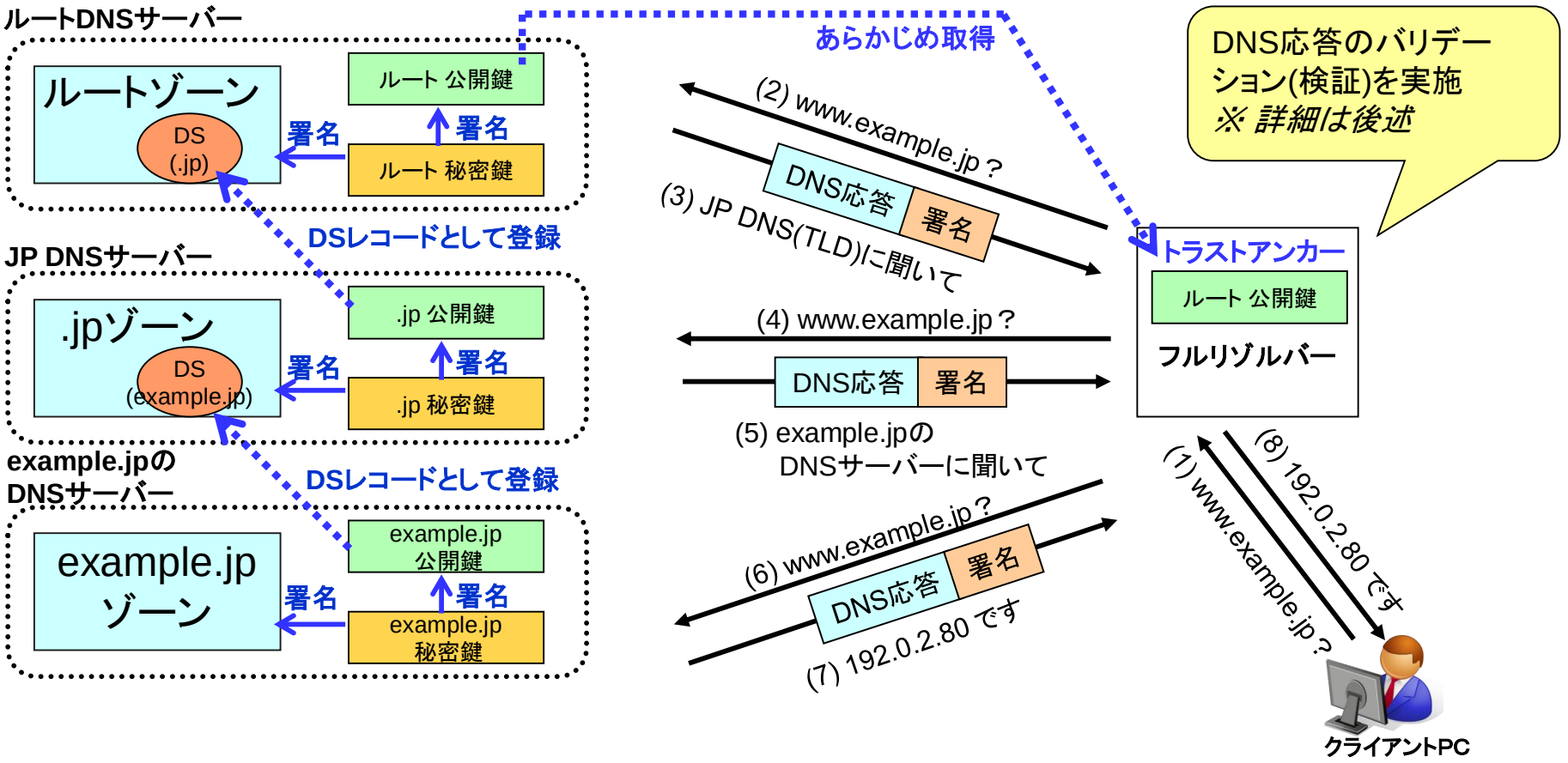
従来のDNSとDNSSECの比較



- 署名検証に失敗した場合、名前解決不能
  - DNSSECは偽の応答を検知する技術
    ⇒ 正しい応答を見つけ出す技術ではない

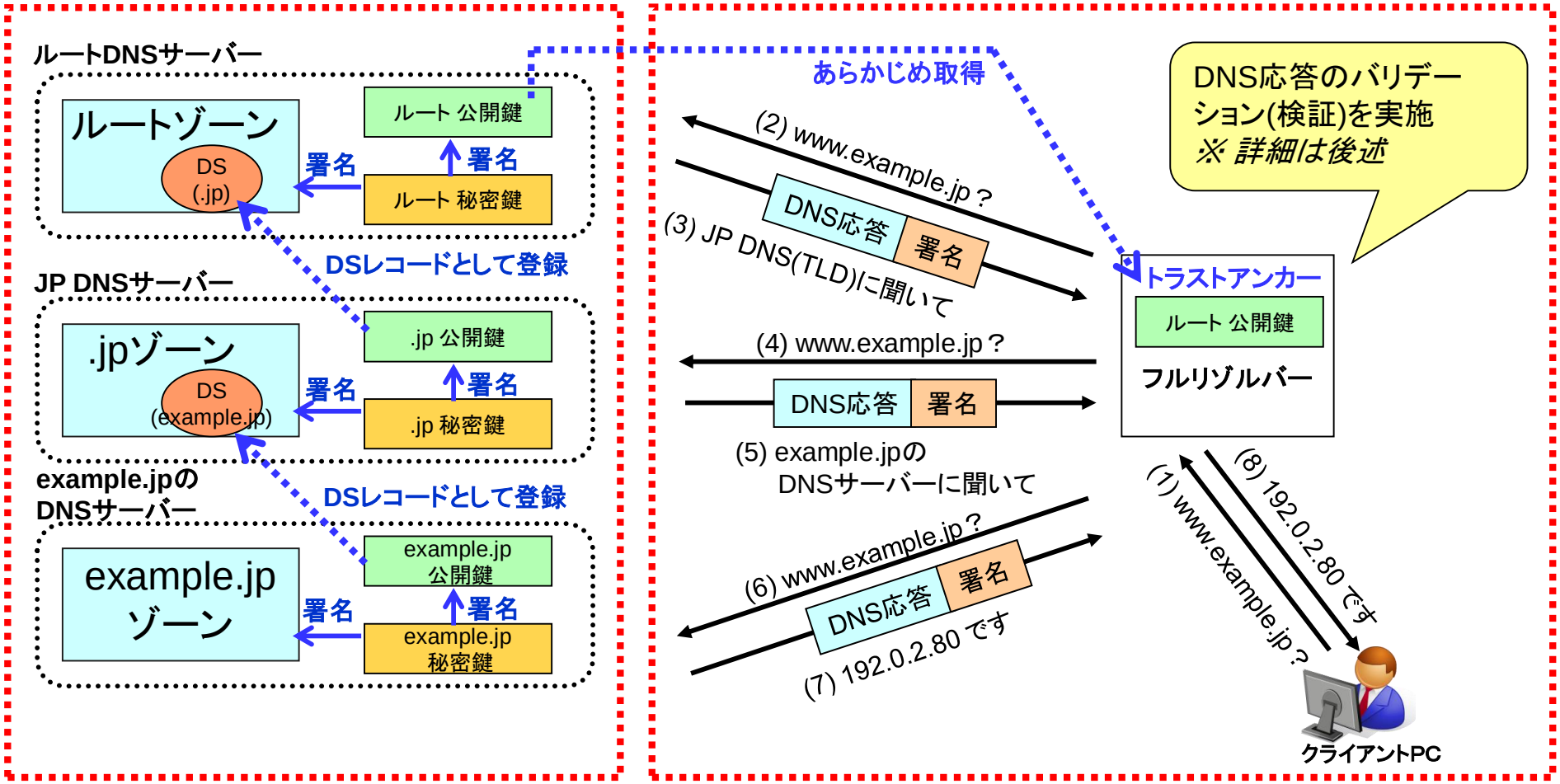
DNSSECの全体像

例：www.example.jp の名前解決の流れ



DNSSECの全体像

例：www.example.jp の名前解決の流れ



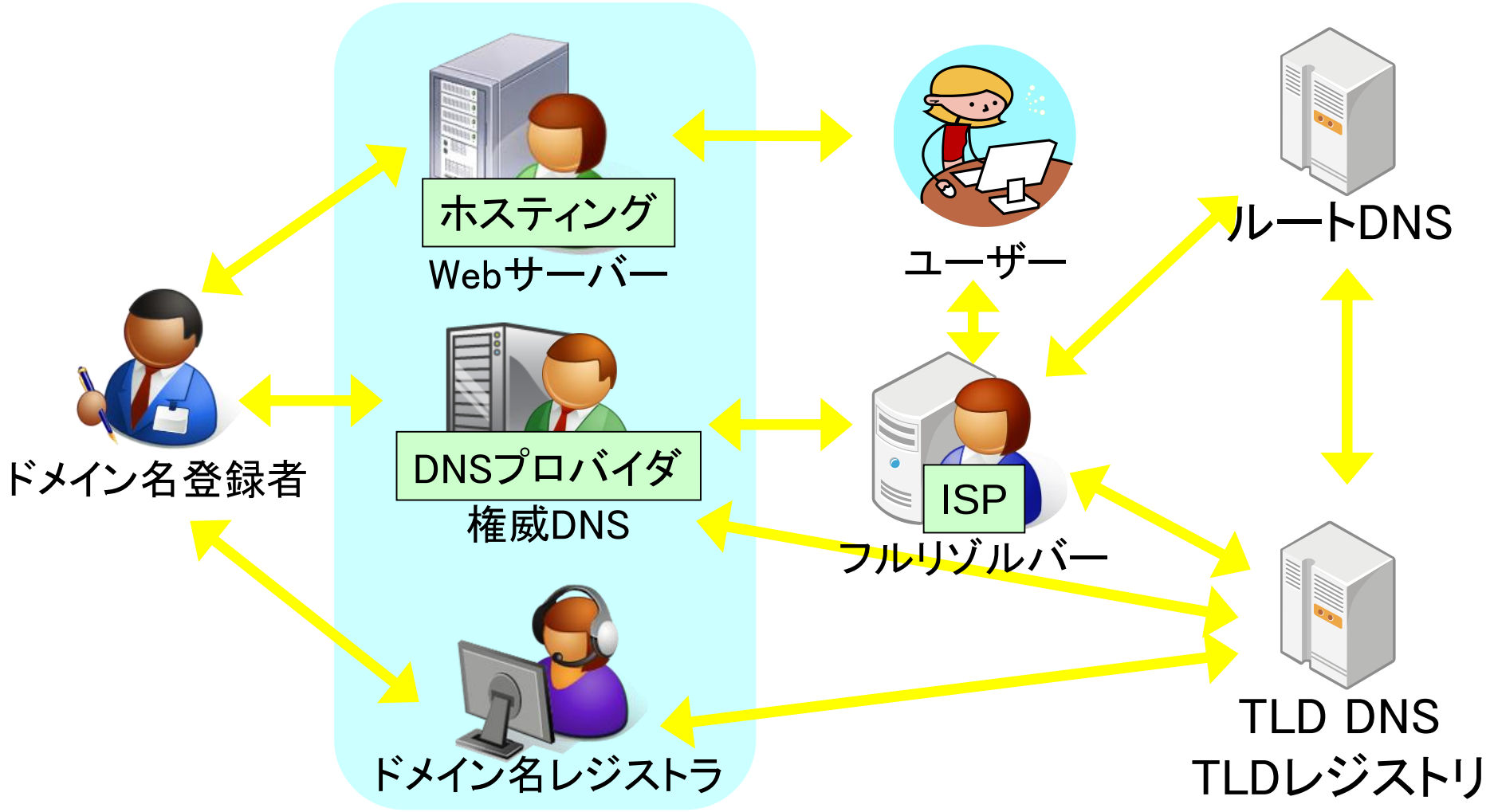
➢ DNSSECの鍵と信頼の連鎖　　➢ DNSSECバリデーションの概要

DNSSEC対応が必要な関係者

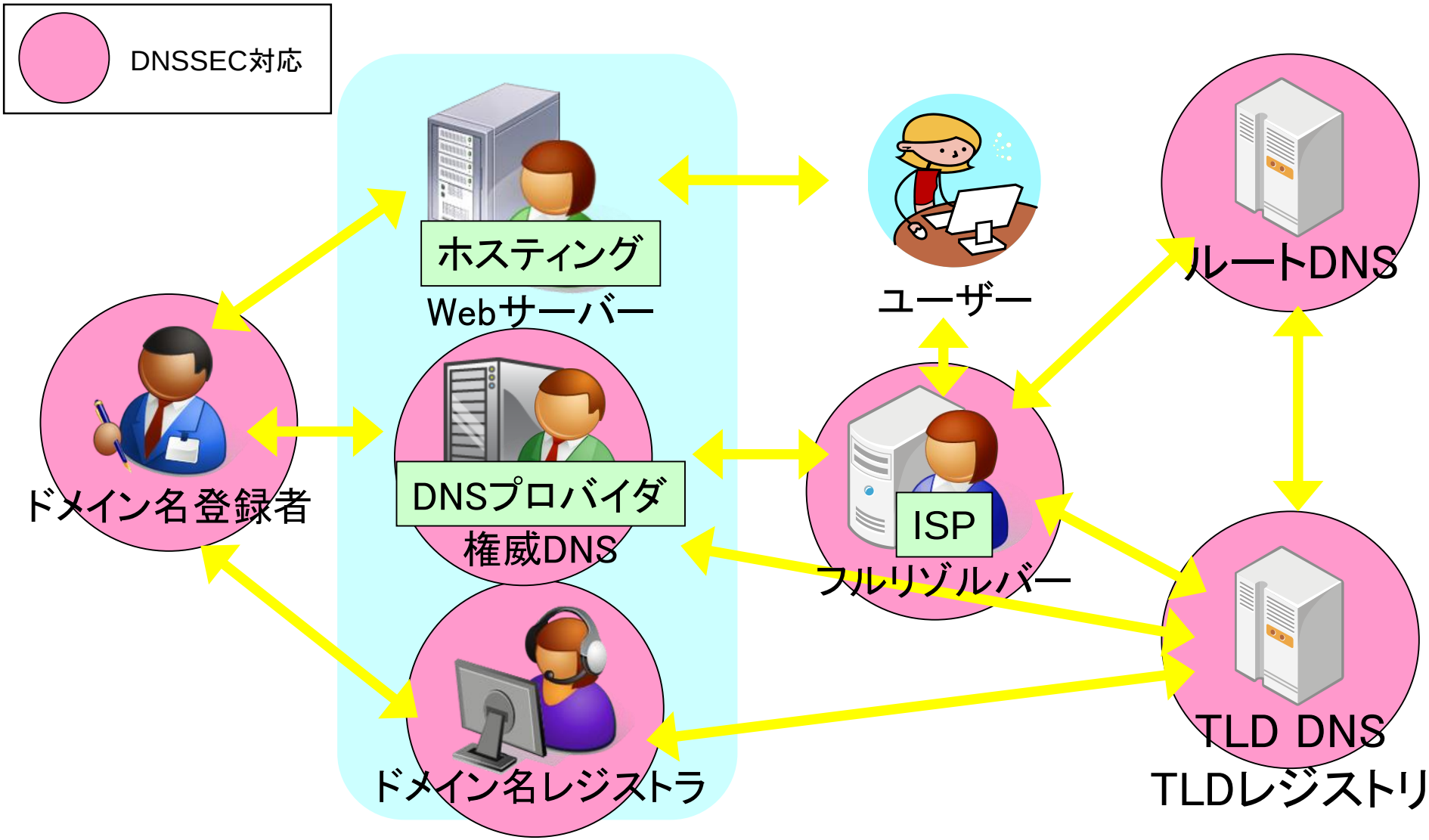
DNSSEC対応作業の概要

- ドメイン名登録者
  - DNSSEC導入の決定
- ドメイン名レジストラ
  - 鍵情報のレジストリへの取り次ぎ
- TLD DNS、ルートDNS
  - 権威DNSサーバーのDNSSEC対応化
  - ゾーンへの署名
- DNSプロバイダ
  - 権威DNSサーバーのDNSSEC対応化
  - 秘密鍵・公開鍵を作成し、ゾーンに署名
- ISP
  - フルリゾルバーのDNSSEC対応化
  - (フルリゾルバーでの)署名の検証

DNS関係者と各情報の流れ



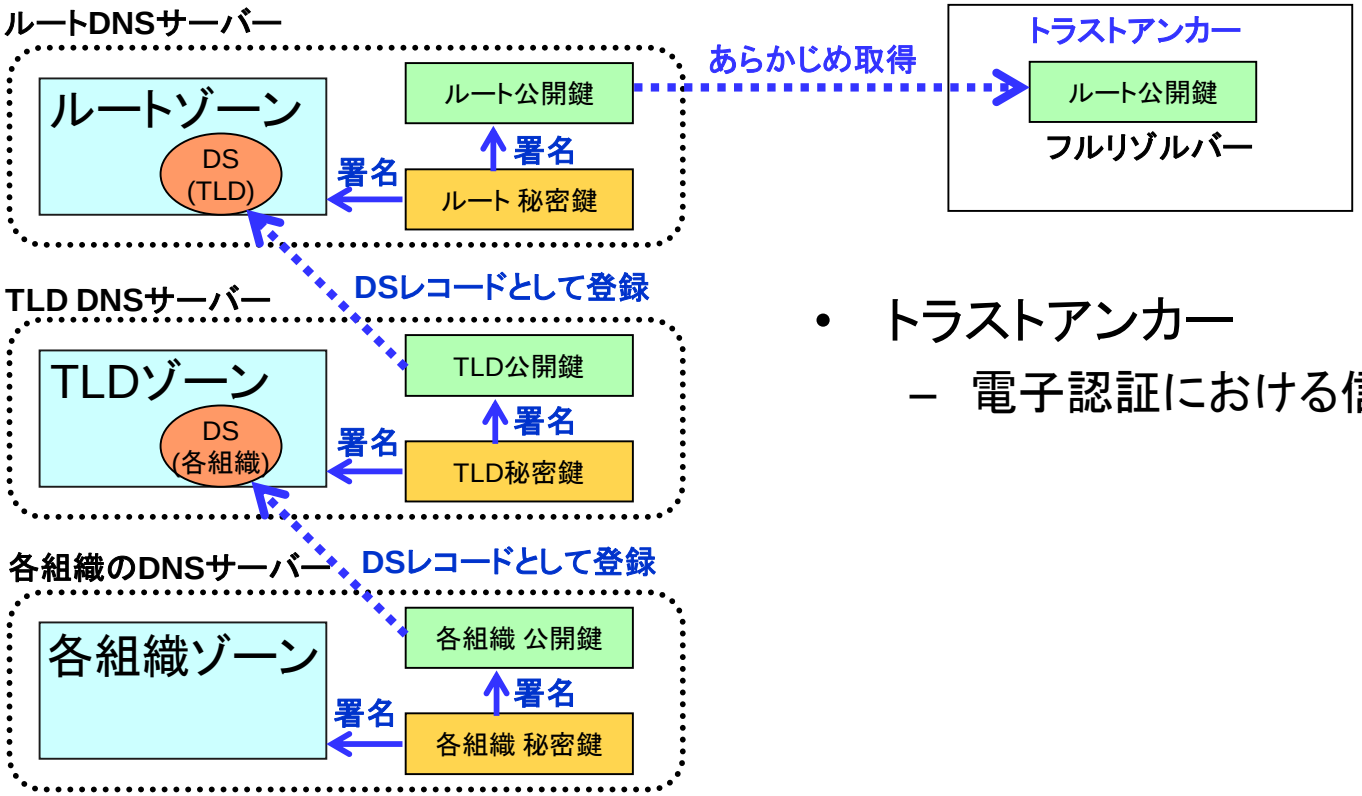
DNSSEC対応が必要な関係者



DNSSECの鍵と信頼の連鎖

DNSSECにおける信頼の連鎖の概念

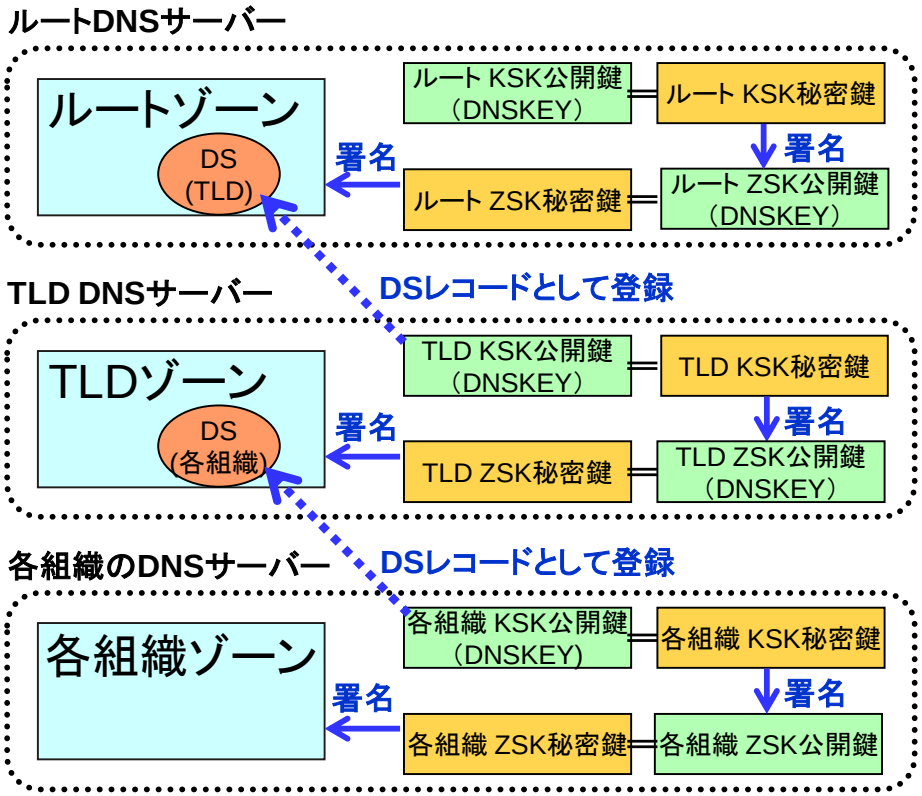
- 秘密鍵で、自ゾーンと子ゾーンの公開鍵に署名
- ルートの公開鍵をフルリゾルバーに登録し、DNSSEC検証を有効にすることで、ルートから各組織ゾーンまでの信頼の連鎖を確立



- トラストアンカー
  - 電子認証における信用の起点

DNSSECで利用する2種類の鍵とDS

- 実際には ZSK と KSK 2種類の鍵を使う



- ZSK（Zone Signing Key）
  - ゾーンデータに署名するための鍵
- KSK（Key Signing Key）
  - そのゾーンの公開鍵に署名するための鍵
- DSレコード（Delegation Signer）
  - KSK公開鍵をハッシュ関数で変換したリソースレコード
- DNSKEY（DNS Public Key）
  - ZSK/KSKの公開鍵を示すリソースレコード

2種類の鍵について

- ZSK
  - 比較的、暗号化強度の低い鍵を用いる ※ .jp ゾーンの場合は1024bit
    - 大規模ゾーンの署名に適応するため、署名コストを低くしている
    - 安全性確保のため、ある程度頻繁に鍵を更新する必要がある ※ .jp ゾーンの場合は月1回
  - 鍵更新は親ゾーンとは関係なく独立して行える
- KSK
  - 比較的、暗号強度の高い鍵を用いる ※ .jp ゾーンの場合は2048bit
    - 署名コストは高いが、ZSK生成時のみ署名を行うため問題にならない
    - 暗号強度が高い分、鍵の更新頻度を少なくすることができる ※ .jp ゾーンの場合は年1回
  - 鍵更新の際はDSを生成し、親ゾーンに登録する必要がある

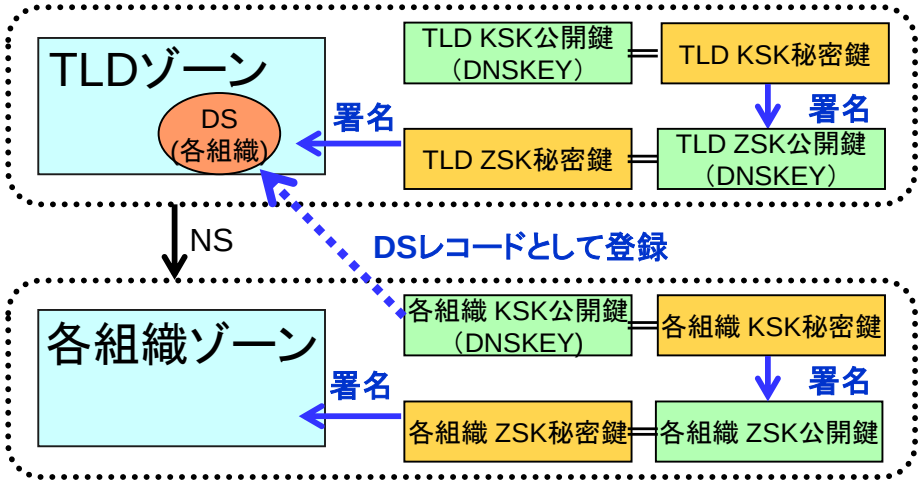
- 2種類の鍵を使い分ける理由

⇒ 鍵の更新による安全性の確保と暗号強度の両立のため

- ZSK: (大規模ゾーンの署名に適応するため)暗号強度を低くしている分、ある程度頻繁に鍵を更新
- KSK: 更新に手間が掛かる分、暗号強度を高くして更新頻度を少なく

DSとDNSKEYレコード

- DS
  - KSK公開鍵を、SHA-1/SHA-256などのハッシュ関数で変換したリソースレコード
    - KSK公開鍵を使用する場合と暗号論的に等価の情報
  - 親ゾーンの委任ポイントに、NSと共に子ゾーンのDSを登録
  - 親ゾーンのZSKでDSに署名することにより、信頼の連鎖を構築可能に
- DNSKEY
  - KSK/ZSK公開鍵の情報を格納するためのリソースレコード



DNSSEC関連のリソースレコード(RR)一覧

- DS　　KSK公開鍵のハッシュ値を含む情報(親ゾーンに登録)
- DNSKEY　　KSK/ZSK公開鍵の情報
- RRSIG　　各RRSetへの署名
- NSEC　　次のRRSet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
- NSEC3PARAM　　NSEC3の生成に必要な情報

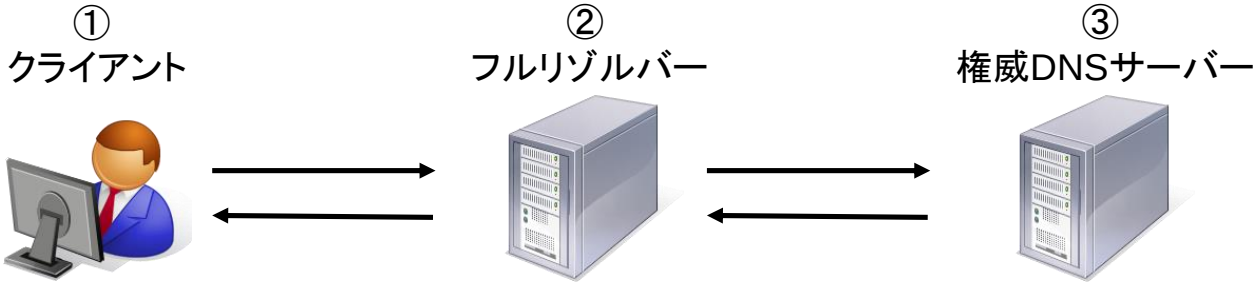
DNSSECバリデーションの概要

用語：バリデーター(Validator)

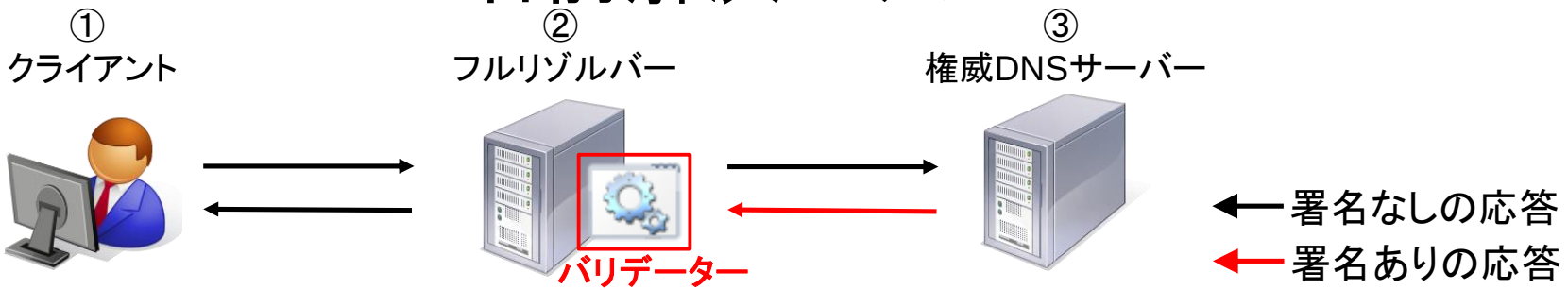
- DNSSECにおいて、署名の検証を行うもの(プログラム、ライブラリ)を指す
- バリデーターの所在
  - フルリゾルバーが署名検証を行う場合、フルリゾルバーがバリデーターとなる
    ⇒ 現状、もっとも一般的なDNSSECのモデル
  - WEBブラウザーなどのDNS検索を行うアプリケーションが直接署名検証を行うモデルも考えられる

DNSSEC対応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



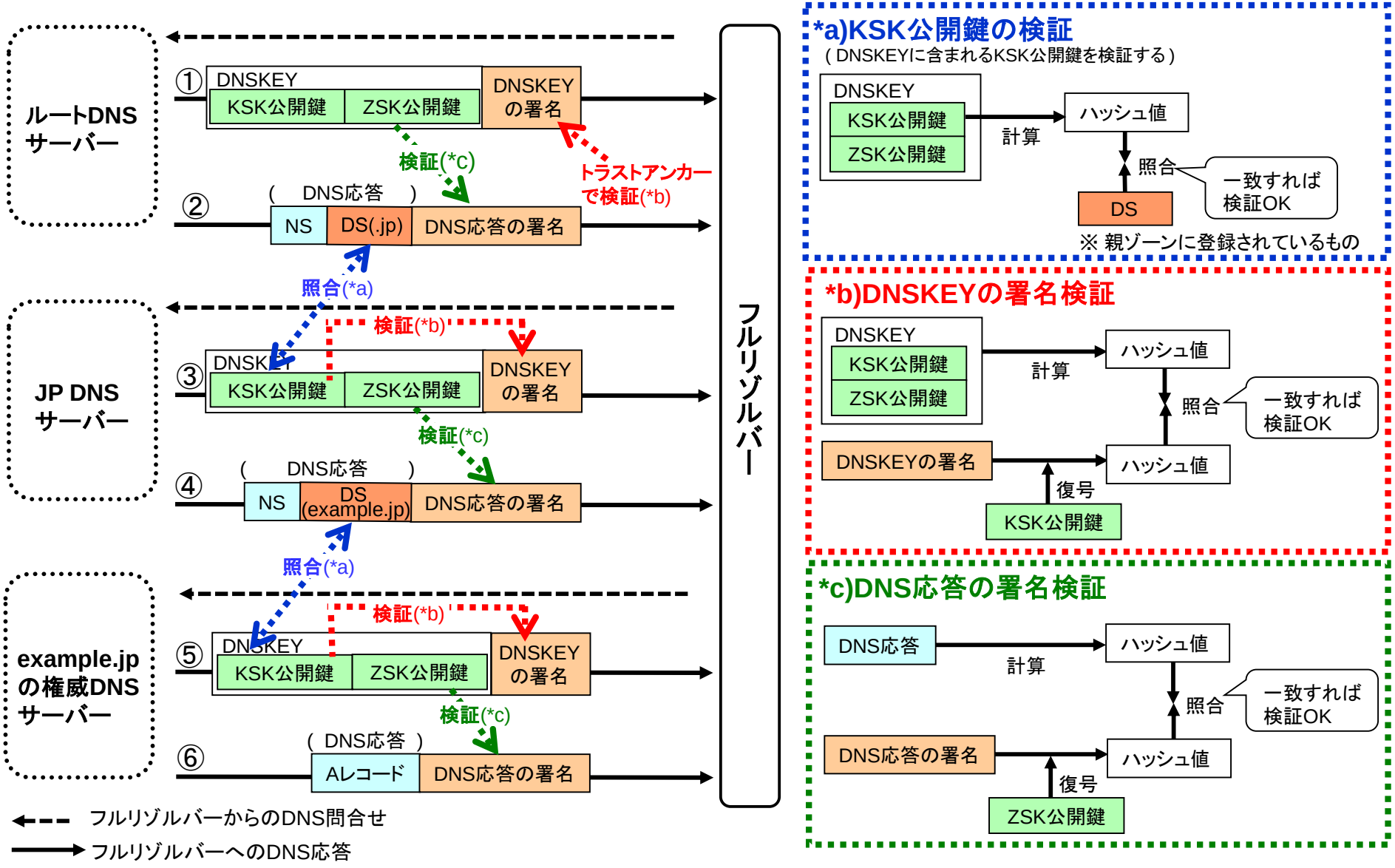
- DNSSECでの名前解決モデル



  - 多くの場合バリデーターは②に実装
  - バリデーターを①に実装することもできる

DNSSEC検証の例

※ フルリゾルバーがバリデーターの場合



digコマンドによる検証方法

$ dig +dnssec jprs.jp *@192.0.2.1(フルリゾルバーのIPアドレス)*

```
; <<>> DiG 9.10.3 <<>> +dnssec jprs.jp
 ;; global options: +cmd
 ;; Got answer:
 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19744
 ;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 11

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

;; ANSWER SECTION:
```

【略】

- 「ad」フラグが付加されていることで、DNSSECの検証に成功したことが分かる

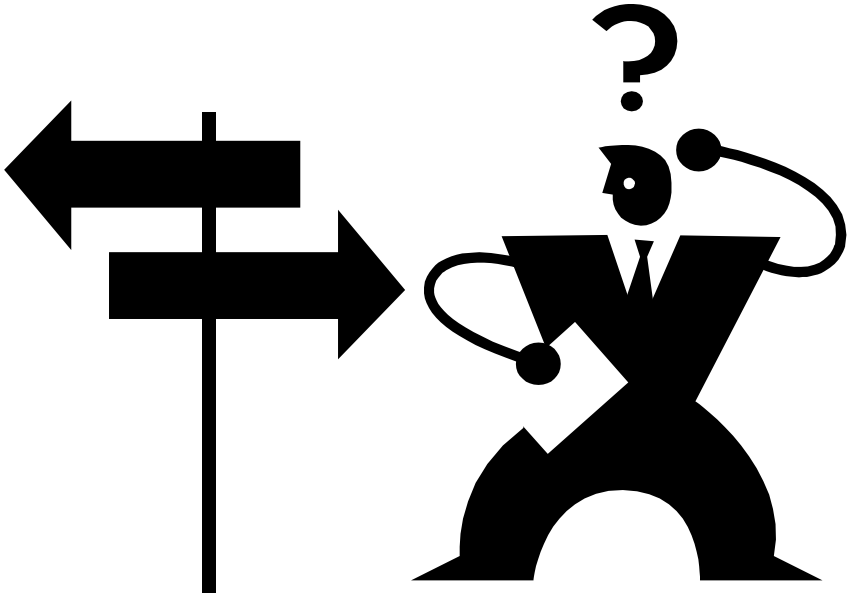
DNSSEC検証に対応する方法(概要)

- DNSSEC対応アプリケーションの導入
  - BINDの場合は9.7系以降が対応
- DNSSEC機能を有効にし、トラストアンカーを設定する
- EDNS0の許可
  - 署名情報などを付加するため、伝統的なDNSのUDPの1パケットに格納できるデータの長さ（512オクテット）を超えてしまう
  - 512オクテットを超えるUDPでの通信を可能にするため、ネットワークの経路上でEDNS0の許可が必要
- サーバーの時刻同期
  - 署名には有効期間があり、正しい署名であってもサーバーの時刻が極端に違うと、署名検証に失敗する
  - NTPなどを利用して、確実な時刻同期を行う

まとめ

- DNSSECは、公開鍵暗号技術を利用した署名によるDNS応答検証のためのしくみ
  - 署名検証に失敗した場合、名前解決不能
    - DNSSECは偽の応答を検知するための技術であり、正しい応答を見つけ出すための技術ではない
  - KSKとZSKの二つの鍵を使う
  - 親ゾーンにはNSに加えてDSを登録する
  - 定期的な鍵の更新と再署名が必要
  - バリデーターは署名の検証を行うもの(プログラム、ライブラリ)を指す
    - フルリゾルバーがバリデーション(署名検証)を行うのが現状、もっとも一般的なDNSSECモデルである
    - WebブラウザーなどのDNS検索を行うアプリケーションが、直接バリデーションを行うモデルも考えられる
    - 署名を行っても検証する仕組みがなければ意味がない
    - バリデーターはDNSSEC対応において必須の存在である

JPRS
JAPAN REGISTRY SERVICES



参考資料:

DNSSECのリソースレコード(RR)

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 (16bit)
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

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  ②③              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アルゴリズム番号一覧

https://www.iana.org/assignments/dns-sec-alg-numbers

RRSIG RR

- 各RRへの署名で、RRSetごとに存在する

```
ns.example.jp. IN RRSIG A 5 3 86400
                         ①②③  ④
    20151218144031 20151118144031 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

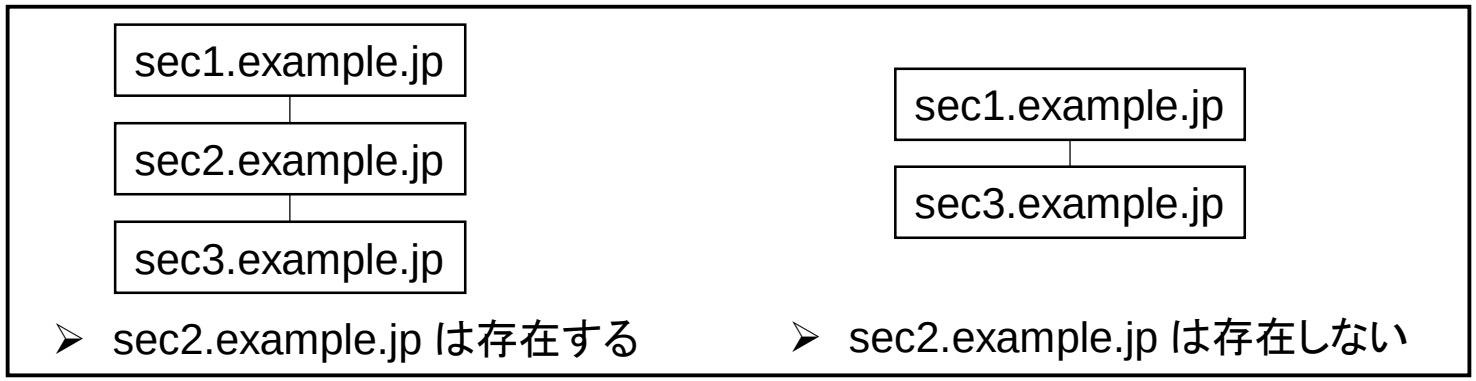
RRSIG RR(続き)

④ 署名対象RRのTTL
⑤ 署名の満了時刻
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のすべて(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

NSEC/NSEC3 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 NSEC3 の違いは後述
- DNSSECにおける不在証明
  - 存在しない名前の偽装を防止するために「存在しない」ことを証明する必要がある



NSEC/NSEC3 RRの違い

- 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入手できる

  ⇒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入手可能
- NSEC3 (RFC 5155)
  – 名前情報の列挙を困難にするための拡張方式であり、ドメイン名を一方向性ハッシュ関数でハッシュ化したものを整列する

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