RFC 5011
DNSSEC.jpプロトコル理解SWG

日本レジストリサービス

Abstract

- DNSSECトラストアンカーの自動アップデート
  - N個の鍵セット中N-1個の鍵が危殆化している場合においても保護される方法を提供
  - 有効なトラストアンカーと同じ階層にトラストアンカーを追加することによって信頼を確立する

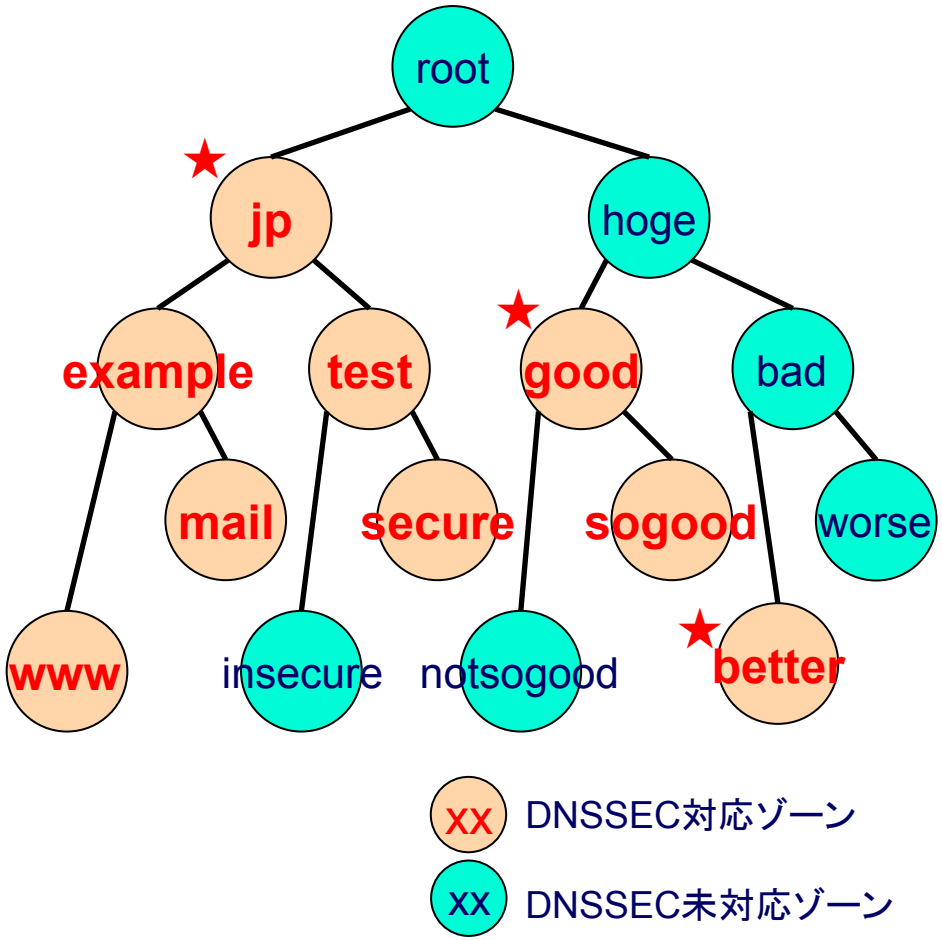
- この機構により要求される変更点
  - リゾルバの管理方法
    - ✓リゾルバの名前解決自体の振る舞いではない
  - DNSKEYレコードへのフラグビット一つを追加

Table of Contents

1. Introduction

- DNSSECは、1つの署名済み名前空間ではなく、DNSの木構造で各々署名済み名前空間の島が存在している
  - それらはトラストポイント名で識別され、少なくとも1つの関連づけられた公開鍵で検証される
  - トラストポイント名に関連する公開鍵を「トラストアンカー」と呼ぶ
    - ✓ 右図の★はトラストポイント
    - ✓ トラストポイントのKSKがトラストアンカー
  - DNSSEC-aware resolverはトラストアンカーを保持すれば、DNSSECで保護された枝の下位ドメインを検証できる
  - 信頼の連鎖をトラストアンカーまで遡ることができるツリーは、「secure」であると考えられる
  - 多数ドメイン名のトラストアンカーを手動で更新するのは問題含みなので、リゾルバが人間の介入なしにトラストアンカーの更新を行う仕組みをここで記述する
  - トラストアンカーの初期設定は議論しない

- 1.1 Compliance Nomenclature
  - 略



2. Theory of Operation

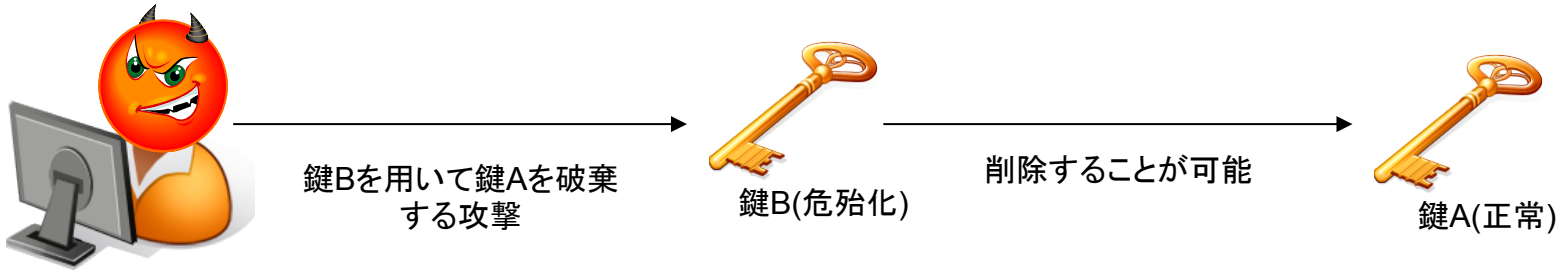
- 仕組みの一般的な概念
  - 既存のトラストアンカーを用いて同じDNS階層の新しいトラストアンカーを認証する
    - ✓ゾーンのオペレータがトラストポイントのDNSKEY RRSetに新しいSEP鍵を置く
      - SEP鍵 = KSK
    - ✓既存のトラストアンカーによってRRSetの有効性を判定する
    - ✓新しい鍵を有効なトラストアンカーとしてリゾルバに追加する
- この方法において解決すべき問題が存在する
  - 問題の例：既存の鍵のうちひとつでも危殆的な場合
    - ✓攻撃者により‘有効’なデータが追加されてしまう可能性がある
    - ✓攻撃者によって新しい鍵が追加され、これを基点に古い鍵をすべて取り消される可能性がある

2.1. Revocation

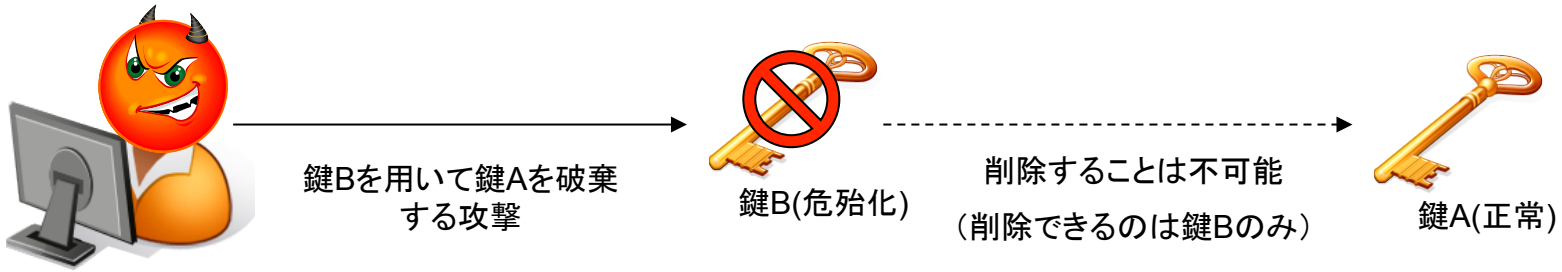
- DNSKEYの秘密鍵を破棄する機構の整備
  - 危殆的な鍵が存在する場合に正常な鍵を無効化(削除)されるのを防ぐ
  - 鍵の破棄はリゾルバが以下を認識した際に行われる
    - ✓ 自己署名されたRRSetの鍵である
    - ✓ REVOKEビットが'1'になっている
  - 自己署名されたRRSetの鍵のREVOKEビットを認識したら、リゾルバは該当の鍵をトラストアンカーとして使用してはならない
    - ✓ ただし破棄のためのRRSIGの検証を除く
    - ✓ 追加とは異なり即時で永続的に破棄される
  - 自己署名されたRRSet
    - ✓ 特別なDNSKEY RRSetではなく検証可能であることを要求するためのもの
    - ✓ 特定のDNSKEYを含む
    - ✓ 検証可能なRRSIGレコードを伴う
- 注意点
  - REVOKEビットがセットされることによりfingerprint(鍵タグ)が変わる
    - ✓ これによりDNSKEYと以下のマッチングに影響がある
      - – 親ゾーンのDSレコード
      - – トラストポイントを利用するリゾルバに保存されているDNSKEYのfingerprint

2.1. Revocation

■ REVOKEビットがない場合



■ REVOKEビットを追加した場合

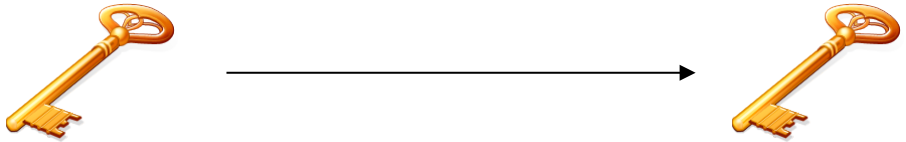


2.2. Add Hold-Down

- トラストアンカー追加においてhold-down timeを導入
  - 攻撃者が(新鍵を追加することで結果として)署名できてしまうリスクを軽減する方法
  - リゾルバが検証可能なトラストポイントのDNSKEY RRSetの新しいKSKを認識したら
    - ✓ タイマーをスタート
    - ✓ 検証されたRRSetの全ての鍵を覚える
  - もしリゾルバが新しい鍵を持たないが有効な署名をされたDNSKEY RRSetを認識したら
    - ✓ この鍵に関する受け入れプロセスを停止
    - ✓ タイマーをリセット
  - もし元々検証に使用されている全ての鍵がタイマーの有効期限までに破棄された場合は
    - ✓ この鍵に関する受け入れプロセスを停止
    - ✓ タイマーをリセット
  - リセットされずにタイマー期限を迎えたら
    - ✓ 新しい鍵はトラストアンカーとして追加される
  - リゾルバは
    - ✓ hold-down time経過後、新しい鍵を含むDNSKEY RRSetを検証するまでは、その鍵をトラストアンカーとして扱ってはならない
- 注意点
  - リゾルバが鍵をトラストアンカーとして受け入れたら、破棄されるまでその鍵をトラストアンカーとして扱わなければならない
  - DNSのもつ分散という特性から、完全な解決策ではない
    - ✓ リゾルバは検索できるものしか知ることができないため
    - ✓ 攻撃者がリゾルバが真のデータを受け取ることを継続的に妨げることができるかもしれない
    - ✓ しかし、このことは現状以下とはならない

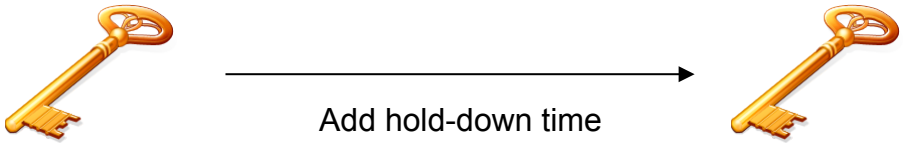
2.2. Add Hold-Down

■ Add hold-downがない場合



→攻撃者が新しい鍵をすぐに利用することができる

■ Add hold-downがある場合



・正当な管理者が毒入れを防げる可能性が上がる

新しい鍵がadd hold-downの間、新旧含む全ての鍵が検証可能でなければ、新しい鍵はトラストアンカーとならない

2.3. Active Refresh

- queryInterval
  - 特定のトラストポイントから鍵の自動更新を行うよう設定しているリゾルバは以下の時間内にトラストポイントについてのクエリを行わなければならない
    - ✓ queryInterval = MAX(1 hr, MIN (15 days, 1/2*OrigTTL, 1/2*RRSigExpirationInterval))
      - – 以下の最小値と1時間のうちの大きい方
        - 15日
        - DNSKEY RRSetのTTLの半分
        - RRSIGが無効になるまでの時間の半分

・負荷を考えれば1時間より短い間隔で問い合わせたくない
・DNSKEYがキャッシュに残っているあいだに再度問い合わせさせたい

- retryTime
  - もしクエリに失敗したら、リゾルバは以下の時間内にクエリを繰り返さなければならない
    - ✓ retryTime = MAX (1 hour, MIN (1 day, .1 * origTTL, .1 * expireInterval))
      - – 以下の最小値と1時間のうちの大きい方
        - 1日
        - DNSKEY RRSetのTTLの10%
        - （RRSIGが）無効になるまでの時間の10%

2.4. Resolver Parameters

2.4.1. Add Hold-Down Time

- 追加についてのhold-down timeは以下の長い方とする
  - ✓ 30 日
  - ✓ 新しい鍵を含んでいると最初に認識したトラストポイントのDNSKEY RRSetのTTLの有効期限
- リゾルバは、新しい鍵を含むDNSKEY RRSetを少なくとも2回、鍵を適用するより前に検証しなければならないことになる

2.4.2. Remove Hold-Down Time

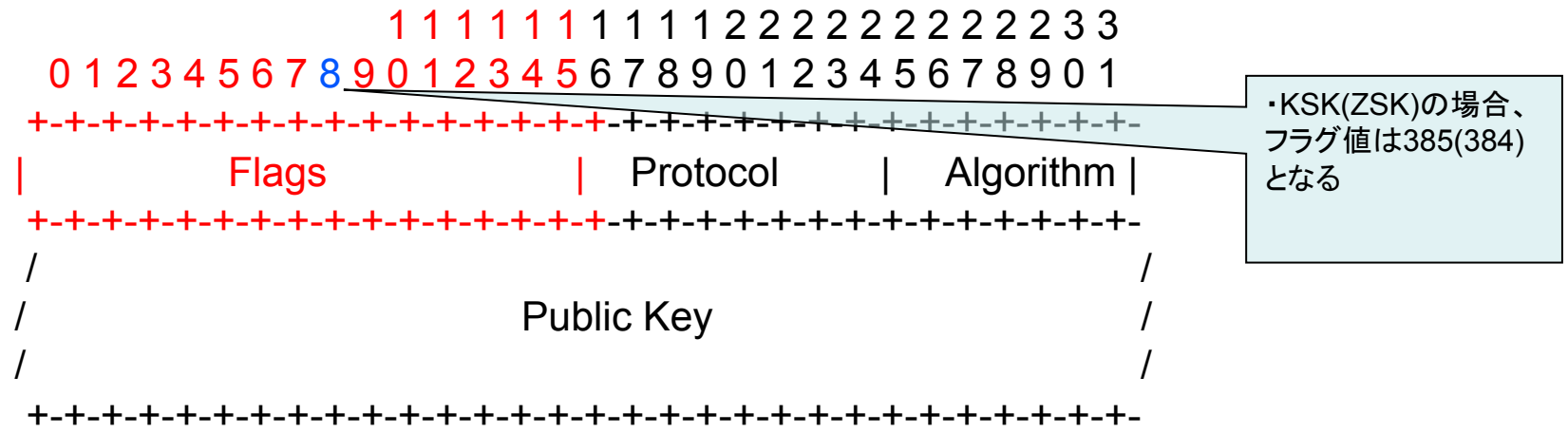
- 削除についてのhold-down timeは30日
  - ✓ 鍵管理データベースのみで利用するパラメータ
  - ✓ データベースから無効な鍵についての情報の削除に失敗しても
    - – 本プロトコルの安全性に影響を与えることはない
    - – ただしデータベースが混乱するかもしれない

2.4.3. Minimum Trust Anchors per Trust Point

- リゾルバはトラストポイントごとに最低5個のKSKを管理できなければならない

3. Changes to DNSKEY RDATA Wire Format
7. IANA Considerations

- DNSKEYフラグフィールドのビット8を‘REVOKE’フラグとして割り振る
  - リゾルバがこれに‘1’がセットされた鍵により署名されたRRSIGを認識した際は
    - ✓ リゾルバはこの鍵の無効化以外の目的でこの鍵を用いた検証を行ってはならない
  - IANAによる割り振り
    - ✓ RFC4034の標準化により割り振れるようになっている

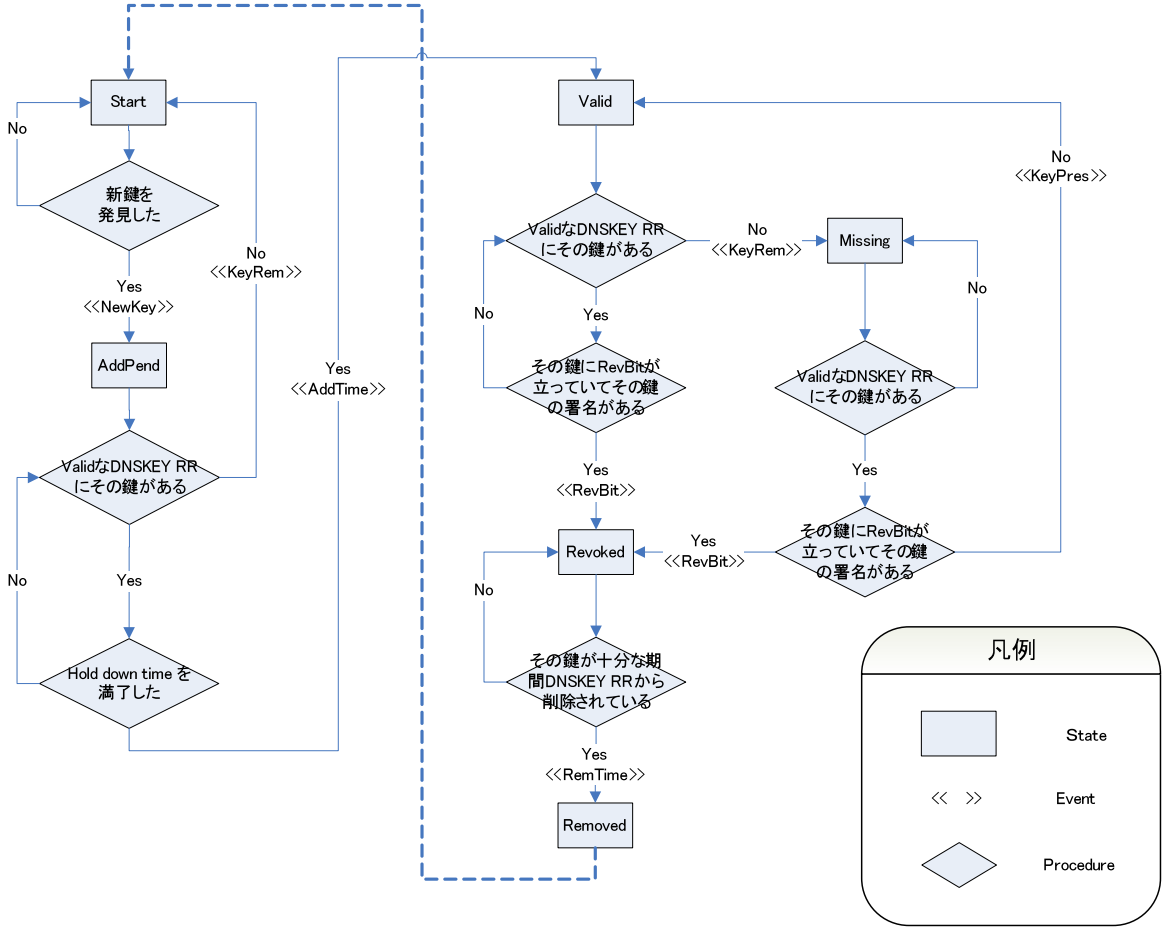


4. State Table

- リゾルバがトラストポイントの鍵をどのように扱うべきかを、鍵の一生のさまざまな状況に応じて記述する
  - 初期状態は‘Start’
  - イベントが発生するとリゾルバにおける鍵の扱いが変化していく
    - ✓ 下表のFROM列が元の状態、NEXT STATEが変化後の状態

	NEXT STATE					
FROM	Start	AddPend	Valid	Missing	Revoked	Removed
Start		NewKey				
AddPend	KeyRem		AddTime			
Valid				KeyRem	Revbit	
Missing			KeyPres		Revbit	
Revoked						RemTime
Removed						

4. State Table



4.1. Events

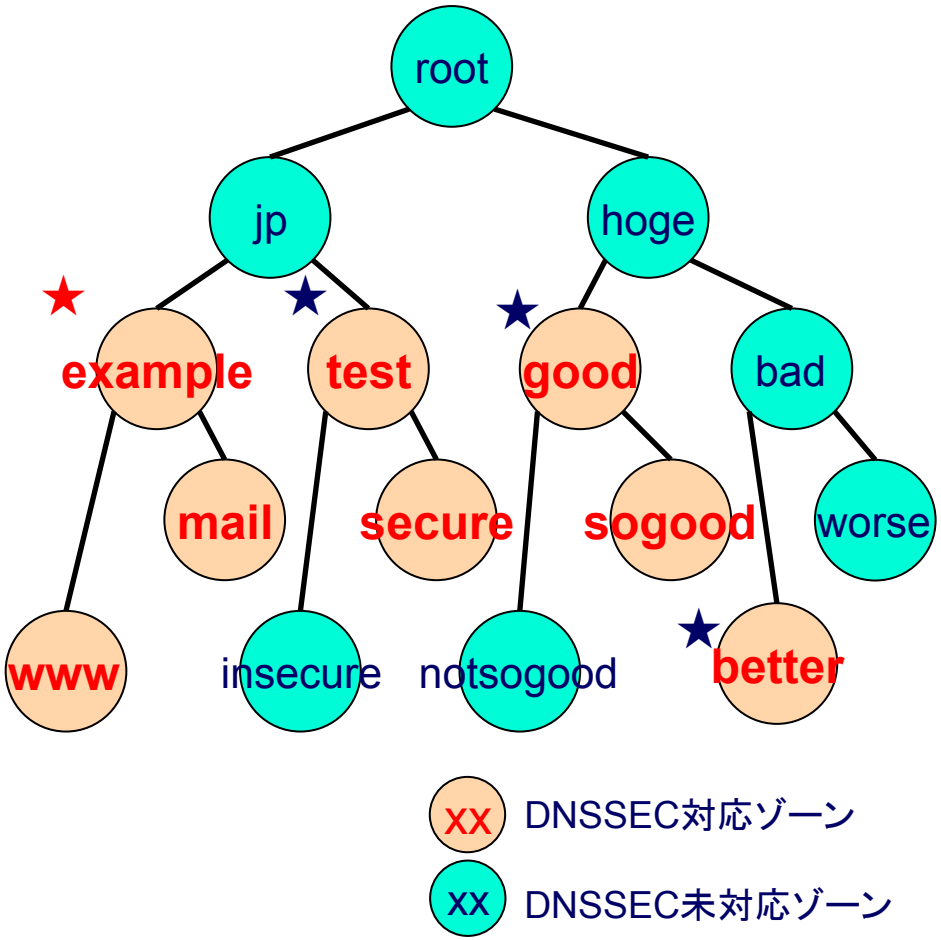
- NewKey
  - リゾルバが新しいKSKを含む有効なDNSKEY RRSetを認識した
  - ‘add time’後もRRSetに存在していたらそのKSKはトラストアンカーとなる
- KeyPres
  - 鍵が有効なDNSKEY RRSetに回復する
- KeyRem
  - リゾルバが当該鍵を含まない有効なDNSKEY RRSetを認識した
- AddTime
  - 当該鍵が少なくとも‘add time’中、全ての有効なDNSKEY RRSetに存在した
- RemTime
  - 取り消された鍵がトラストポイントのDNSKEY RRSetに存在しなくなってから十分な時間が経過した
- RevBit
  - REVOKEビットが立った鍵がトラストアンカーのDNSKEY RRSetに出現し、この鍵自身による有効な署名が存在する

4.2 States

- Start
  - トラストアンカーとなる新しい鍵がリゾルバにまだ認識されていない
  - (ゾーンサーバにあるなしに関わらず)リゾルバの直前の検索では新しい鍵が見つからなかった
- AddPend
  - 'SEP'ビットが立ち、有効なDNSKEY RRSetに含まれる新しい鍵がリゾルバに認識された
  - 鍵がトラストアンカーとして使われる前に hold-down time が存在する
- Valid
  - 鍵がリゾルバにトラストアンカーとして認識された
  - 鍵がhold-down timeの最初からずっと、すべての有効なDNSKEY RRSetに含まれている
    - ✓ DNSKEY RRSetはリゾルバ中に連続して存在する必要はない
- Missing
  - 異常な状態
  - トラストポイントに有効な鍵として残っているはずだが、リゾルバの最新の検証ではDNSKEY RRSetに存在しなかった
  - ゾーンオペレータがREVOKEビットを使用せずに鍵を削除した場合に発生する
- Revoked
  - REVOKEビットが立った鍵でDNSKEY RRSetが署名され、リゾルバに認識された
  - 鍵はトラストアンカーとして恒久的に無効として扱わなければならない
- Removed
  - 十分長いhold-down timeの経過後、リゾルバから鍵が削除された状態
  - トラストアンカーとして有効と扱ってはならない

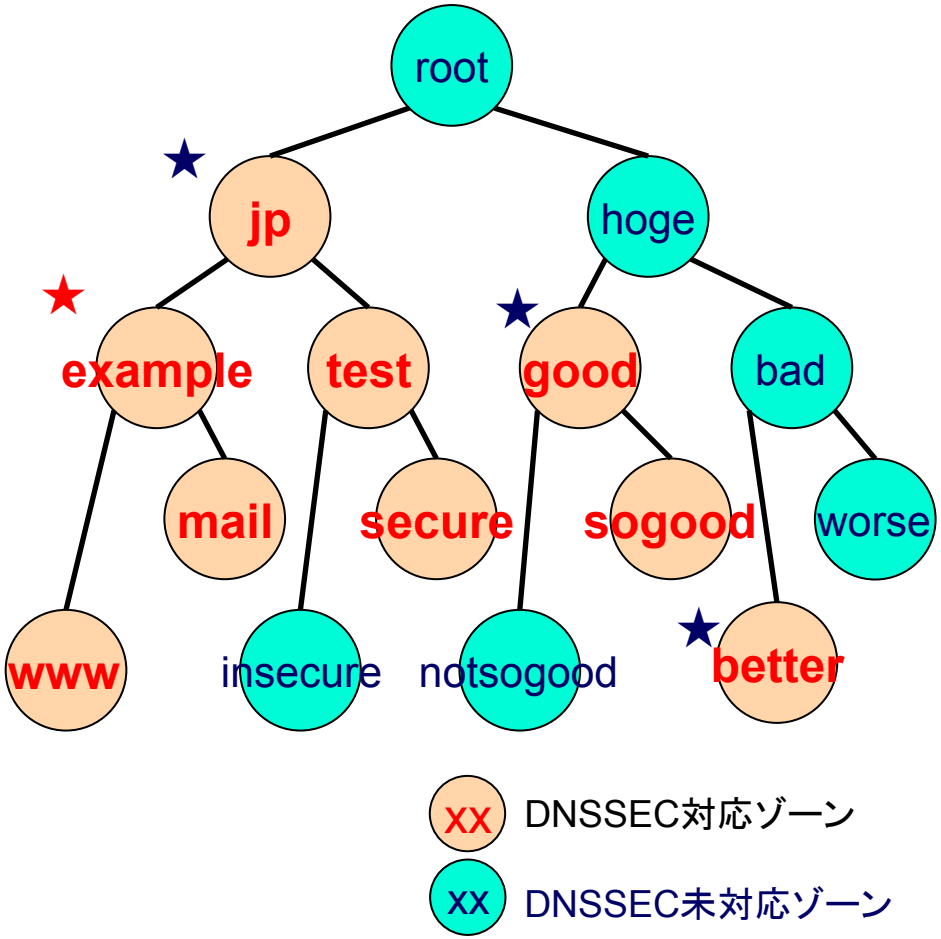
5. Trust Point Deletion

- ★印はトラストポイント
- ★印のトラストポイントを削除する
- トラストポイントを構成するトラストアンカーがすべてrevokeされた場合、そのトラストポイントは最初から存在しなかったものと扱う
- 上位のトラストポイントが存在しない場合は、削除されたトラストポイントから下位に存在するデータはinsecureと判断する
  - example.jp 以下は、insecureとなる



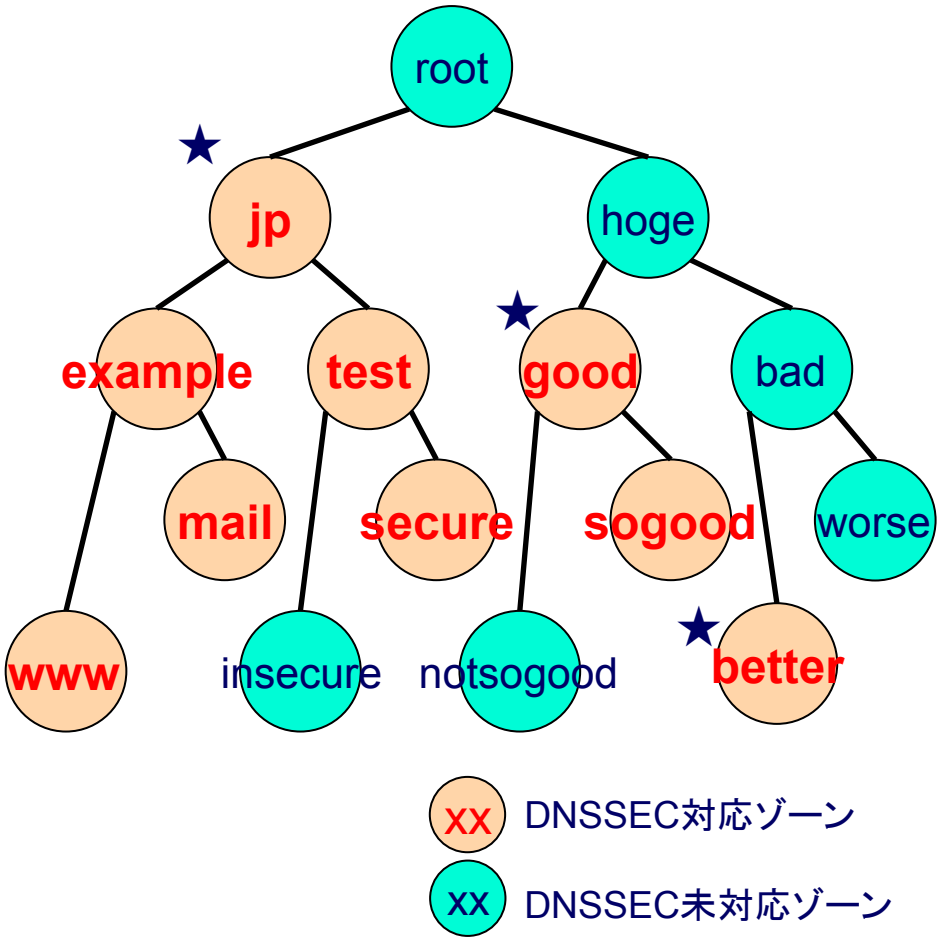
5. Trust Point Deletion

- ★印のトラストポイントを削除する
- 上位のトラストポイントが存在する場合は、削除されたトラストポイントから下位に存在するデータは上位のトラストポイントにより評価される
  - この場合は、jpのトラストポイントから信頼の連鎖が存在する
- 他のトラストポイントにより信頼の連鎖が構成された下位のトラストポイントは180日後にresolverから削除してもよい
- 下位のトラストポイントを削除するかはローカルな判断による



6. Scenarios - Informative

- オペレーションにおいて推奨されるモデル
  - Active key 1つとStand-by key 1つを各トラストポイントに持つ
    - ✓ 右図の★に持つことになる
  - Active key
    - ✓ DNSKEY RRSetの署名に使用される鍵
  - Stand-by key
    - ✓ 通常はRRSetの署名に使用されない
    - ✓ ただしリゾルバはトラストアンカーとして受け入れる
  - Stand-by Keyは署名に使用しないので、その秘密鍵はActive keyの場合より厳重に保護できるはず
    - ✓ 概念上Stand-by keyはActive Keyより危殆的なものでないがここでは議論しない



6.1. Adding a Trust Anchor

■ 仮定

- 既存のトラストアンカーの鍵を‘A’とする

手順	トラストアンカーの追加
1	新しい鍵ペアを生成する
2	鍵ペアからDNSKEYレコードを作りフラグを257にする
3	DNSKEYをRRSetに追加する
4	存在するトラストアンカーの鍵‘A’のみでDNSKEY RRSetを署名する
5	リゾルバが新しいDNSKEY RRSetとその署名を検索するのを待つ
6	新しいトラストアンカーが‘State Table’の方式によってリゾルバに行き渡る

6.2. Deleting a Trust Anchor

■ 仮定

- ➢ 既存のトラストアンカーを‘A’と‘B’とする
  - ✓ ‘A’を破棄して削除したいものとする

手順	トラストアンカーの削除
1	REVOKEビットを‘A’にセットする
2	■‘A’と‘B’の両方を用いてDNSKEY RRSetを署名する ➢‘A’が即時に破棄される ➢ゾーンオペレータは破棄されたRRSet中の‘A’を最低でも削除のhold-down timeまで保持しなければならない ✓hold-down time後にDNSKEY RRSetから削除できる

6.3. Key Roll-Over
6.4. Active Key Compromised
6.5. Stand-by Key Compromised

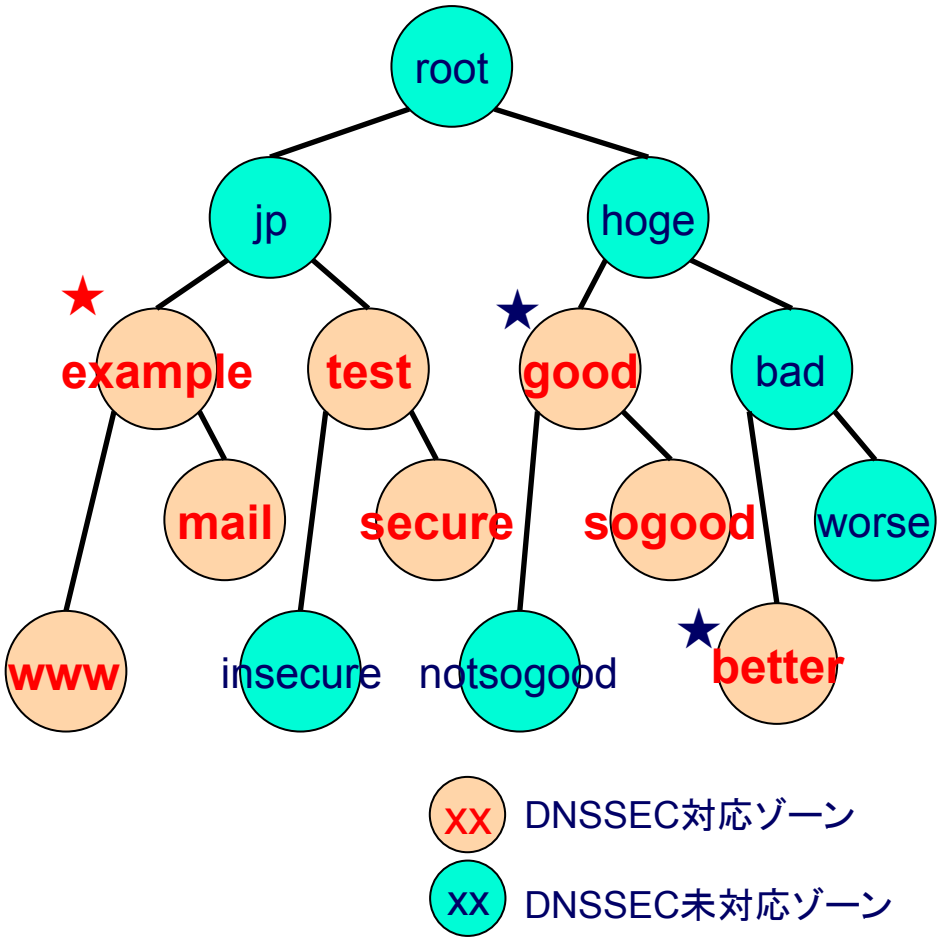
■ 仮定

- ➢ 鍵‘A’=Active Key
  - ✓ DNSKEY RRSetに署名している
- ➢ 鍵‘B’=Stand-by Key
  - ✓ DNSKEY RRSetに存在しトラストアンカーと認識されているが署名していない

手順	通常の交換(6.3)または’A’の危殆時の対応(6.4)	‘B’の危殆時の対応(6.5)
1.	新しい鍵ペア’C’を生成	
2.	‘C’をDNSKEY RRSetに追加	
3.	‘A’にREVOKEビットをセット (5011に従うなら必ずRevoke Bitを立てる)	‘B’にREVOKEビットをセット
4.	‘A’と’B’を用いてRRSetを署名	
結果	■‘A’→即時に破棄される ➢オペレータは’A’についてのRRSetをhold-down timeまで削除すべきでない ■‘B’→即時にActive Keyとなる ■‘C’→hold-downの期限切れ後Stand-by Keyとなる	■‘A’→Active Keyとして残る ■‘B’→即時に破棄される ➢‘B’についてのRRSetは削除のhold-down timeまで残すべきである ■‘C’→hold-downの期限切れ後Stand-by Keyとなる

6.6. Trust Point Deletion

- ★印はトラストポイント
- ★印のトラストポイントを削除する
- 手順
  1. example.jpゾーンの新しいDNSKEYとDSレコードを生成し、古い鍵のDSレコードと共に新しい鍵のDSレコードを上位のjpゾーンに登録する
  2. jpゾーンにDSが登録されたら、新旧全ての鍵で署名することにより、新しいDNSKEYをRRSetに追加するとともに全ての古い鍵を破棄する
  3. 30日後、古い鍵の登録を止め、破棄する鍵とそのDSレコードをjpゾーンから削除する
- トラストポイントにおいて古い鍵の破棄と新しい鍵の追加を同時に行うことにより、リゾルバに新しい鍵がトラストアンカーとして追加されるのを防ぐ
- 古い鍵のDSレコードを削除するのみだと、信頼の連鎖が途切れてしまう



8. Security Considerations
8.1. Key Ownership vs. Acceptance Policy

- 8. Security Considerations
  - Section 2に関連した事項
    - ✓ 特にSection 2.2.
      - 本プロトコルに特有でないセキュリティ上の問題はRFC4986に記述がある
- 8.1. Key Ownership vs. Acceptance Policy
  - ゾーン所有者は鍵の作成と配布に責任を持っているが、ゾーンの情報として鍵を受け入れるかどうかはリゾルバ所有者が決定する
    - ✓ 本実装を受け入れるかはリゾルバの所有者の決定による
    - ✓ リゾルバの所有者や実装者はこの仕組みを特定のトラストポイントに用いて、鍵更新の許可・不可を選択してもよい
      - 不可とする場合は更新の手法を確立する必要があるが、これについては本文書の範囲外とする

8.2. Multiple Key Compromise
8.3. Dynamic Updates

8.2. Multiple Key Compromise

- 本方式では、少なくとも1つの検証可能なトラストアンカーの鍵が危殆的でなれば状況回復を行うことができる
- ゾーンの所有者は、検証可能なトラストアンカーの登録数を定め、危殆的な状況を検知した際の回復手順を用意すべきである
- 全ての鍵が危殆的な場合は、すべてのリゾルバを手動または他の方法で更新する方法が必要となる

8.3 Dynamic Updates

- in-band keyの情報を元にしてリゾルバにトラストアンカーの更新を許可するのは手動更新より安全ではない
- しかし、トラストアンカーが危殆化すると数多くのリゾルバの更新が必要となること、その際の標準的管理手法が存在しないことを考えれば、このアプローチは現状ほど悪くない

9. Normative References
10. Informative References

- 9. Normative References
  - RFC2119
  - RFC3755
    - ✓ Legacy Resolver Compatibility for Delegation Signer (DS)
      - DSに対する旧式のリゾルバへの互換性
  - RFC4033
  - RFC4034
  - RFC4035
- 10. Informative References
  - RFC4986
    - ✓ Requirements Related to DNS Security (DNSSEC) Trust Anchor Rollover
      - DNSSECトラストアンカーのロールオーバーに関する要件