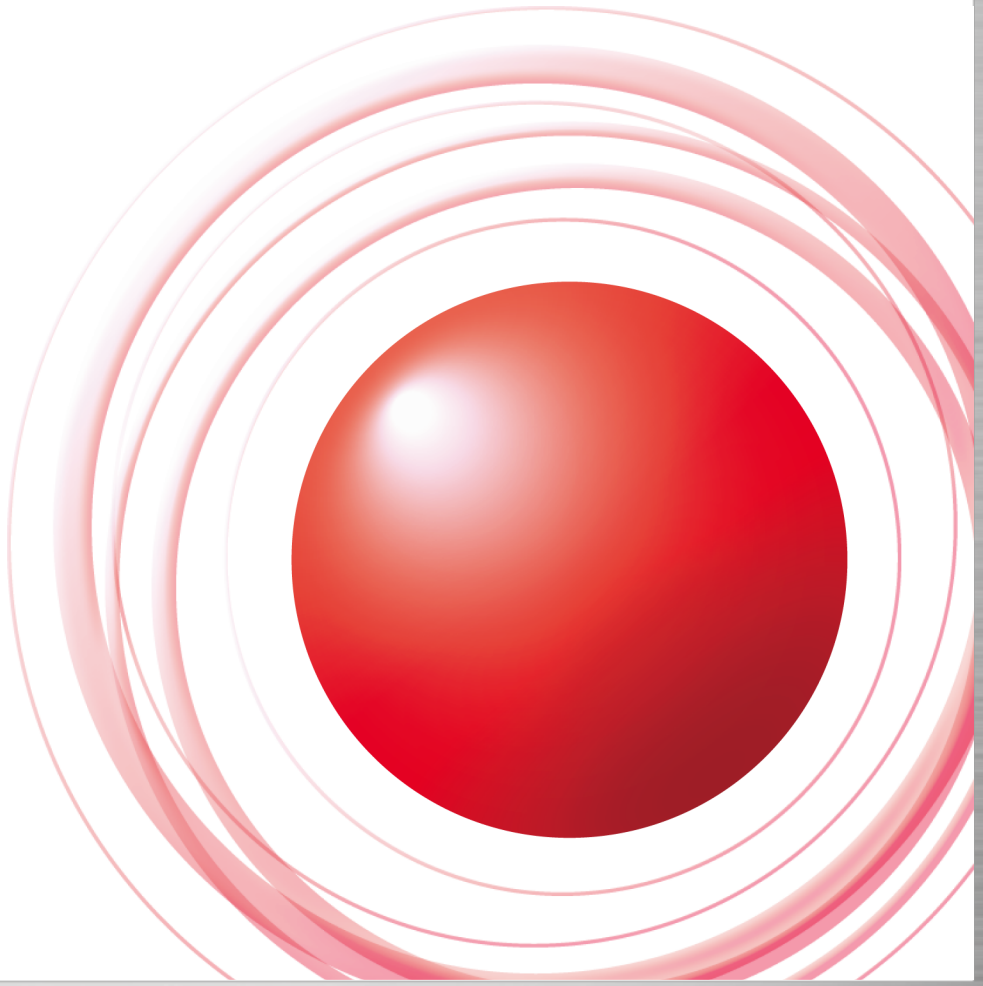
# 権威DNSのDNSSEC対応

IIJ
Internet Initiative Japan



2012/11/21
株式会社インターネットイニシアティブ
山口崇徳

Ongoing Innovation

# アジェンダ

- **基礎編**
  - DNSSECにおける権威DNSの役割
  - 署名の更新
  - 鍵のロールオーバー
  - パラメータの設計
- **実践編**
  - サーバの構成検討
  - 運用支援ツール
  - 実際にDNSSEC対応させるまでの流れ
  - トラブルシューティング

# 基礎編

# DNSSECにおける権威サーバの役割

- 署名鍵(DNSKEY)の公開
  - KSK、ZSKの2種類
- 署名(RRSIG)の公開
  - 問い合わせに対し、それに対応した署名を応答する
- 不在証明(NSEC/NSEC3)
  - 「存在しない」という応答にお墨つきを与える
- 下位ゾーン委譲
  - NSレコードに加え、下位ゾーンから送られてきたKSKのハッシュ(DS)に署名して公開し、信頼の連鎖を形成する

# 日々のDNSSEC運用

- 大きくわけて3つ
  - 署名の更新
  - 署名鍵の定期的な更新(ロールオーバー)
  - 下位ゾーンのDSレコードをゾーンに載せる
    - 下位ゾーンがなければもちろん不要
- たったこれだけ
  - そんなに仕事が多いわけではない……はず

# 署名の更新

- ゾーン内の各レコードに対応するRRSIGの生成
- 不在証明(NSEC/NSEC3)の生成

- ゾーンを編集したらかならず再署名
- ゾーンを編集しなくても定期的に再署名
  - 署名(RRSIG)には有効期限が設定されているので、無効になる前に期限を再設定する
    - 厳密には期限(いつまで)ではなく期間(いつからいつまで)
  - 変更しないかぎりずっと放置していた従来のDNSとは異なる

# 署名の更新(余談)

- **「管理者が美しい旋律を奏でながら楽しげにゾーンに署名するイベント」**
  - RFC4641におけるゾーンファイルへの署名に関する定義

  *Singing the zone file: The term used for the event where an administrator joyfully signs its zone file while producing melodic sound patterns.*
- **めんどくせーなー、とか思いながらしぶしぶ署名するのは、RFCに反する行為です!**

# 鍵の更新(ロールオーバー)

- 同じ鍵を長期間使い続けることの危険性
  - 暗号を解読された
  - 何らかの事故で秘密鍵が盗まれた、漏洩した
  - 暗号アルゴリズムが危殆化した
- このリスクを低減するため、鍵を更新する
  - 上記のような疑いが生じないかぎり更新しない、という判断もあり
  - ただし、ロールオーバーはかなり煩雑な作業ので、いざ必要になったときに慣れないことを緊急におこなって失敗する危険性が大きい
- 定期的なロールオーバーを推奨
- DNSSECだから鍵更新が必要なわけではない
  - SSLやSSHやPGPその他でも事情は同じ

# ロールオーバーの留意事項

- **DNSには、キャッシュがある**
  - 権威DNS上に置いてあるゾーンからは削除されたとしても、TTLが過ぎてないうちは世間のどこかのキャッシュサーバ上で生きている(可能性がある)
  - 鍵を更新するときは更新前の情報のキャッシュとの整合性まで気を配らなければならない
    - 新しい鍵で署名しても、古い鍵による署名(RRSIG)のキャッシュが有効なうちは、旧鍵をゾーンから削除してはならない、など
- **鍵の更新は、いくつかの段階を踏んでおこなう**
  - DNSSECでいちばんめんどくさいところ

# 鍵の状態遷移

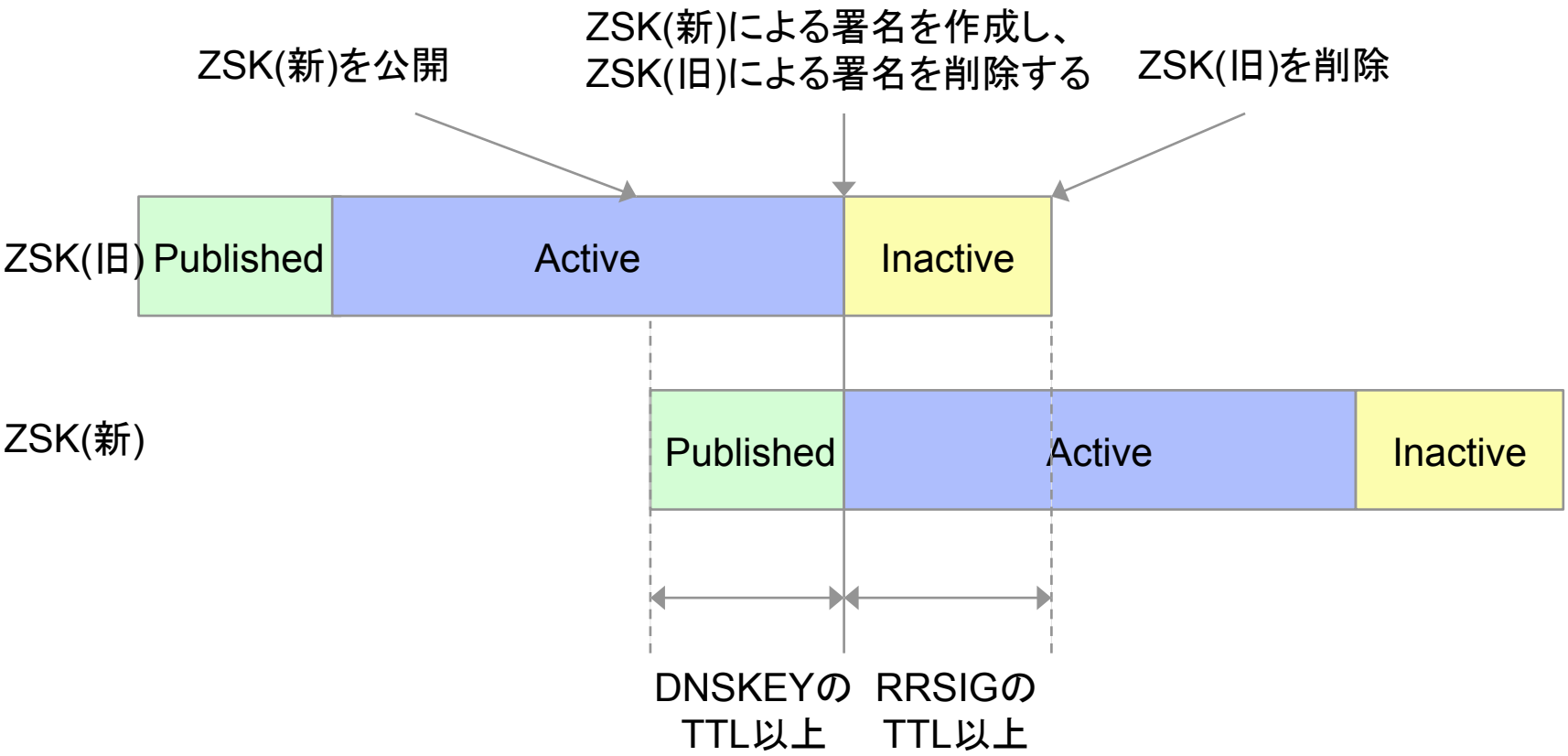
- 大きくわけると3つ
- Published
  - 鍵がゾーンファイルで公開されているが、署名(検証)には使われていない状態
- Active
  - 公開され、署名(検証)に使われている状態
- Inactive
  - 署名(検証)には使わなくなったが、まだ鍵がゾーンに残っている状態
- これらの状態は、DNSSECのプロトコルで規定されているわけではない
  - こういうふうに考えるとわかりやすいよ、という運用上の概念
  - 一見して区別がつくとはかぎらないので、どの鍵が今どの状態にあるのかは、自分で管理しておく必要がある

# ZSKロールオーバー

- **事前公開法(Pre-Publication Method)**
  - 新しい鍵を、その鍵による署名に先がけて公開する

1. **新しい鍵を生成してゾーンに載せる**
   - まだ署名には使わない
2. **DNSKEYのTTL以上の時間が経過して、新旧どちらの鍵もキャッシュサーバから見えるようになるのを待つ**
3. **署名に使う鍵を新鍵に切り替え**
   - 旧鍵での署名は止めるが、ゾーンには残しておく
4. **RRSIGのTTL以上の時間が経過して、旧鍵による署名がキャッシュから消えるのを待つ**
5. **旧鍵を削除**

# 事前公開法によるZSKロールオーバー



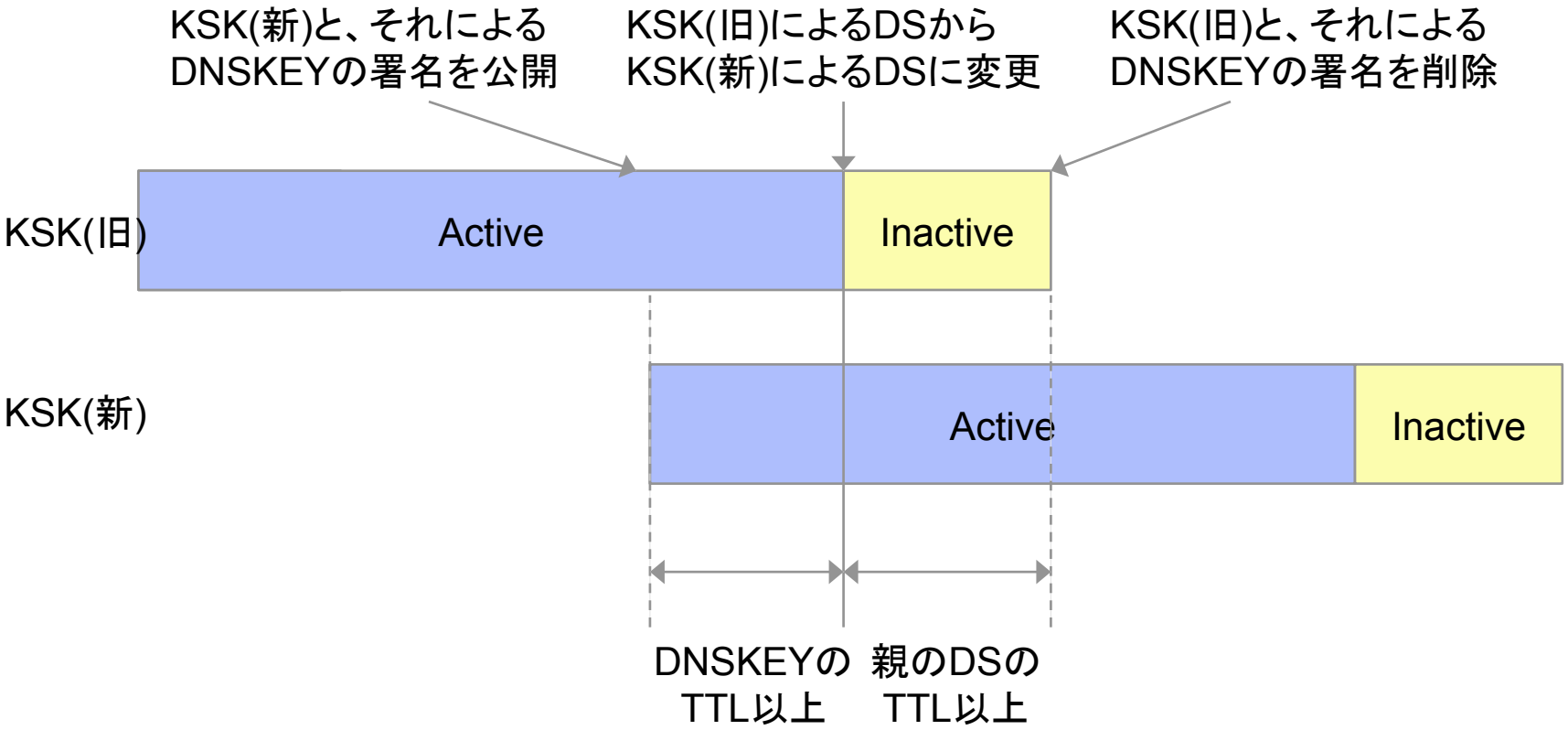

- ZSK(新)の公開と、その鍵で署名を開始するタイミングが異なることに注意

# KSKロールオーバー

- 二重署名法(Double-Signature Method)
  - あるレコードを2つの鍵で別々に署名する

1. 新鍵を公開し、新旧両方の鍵で署名する
2. DNSKEYのTTL以上の時間が経過して、新旧どちらの鍵もキャッシュサーバから見えるようになるのを待つ
3. 親ゾーンに登録するDSを新鍵のものに切り替える
4. DSのTTL以上の時間が経過して旧DSがキャッシュから消えるのを待つ
5. 旧鍵(とそれによる署名)を削除する

# 二重署名法によるKSKロールオーバー




- KSK(新)の公開と同時に、その鍵を使った署名もおこなうが、それが実際に検証に使われるようになるのはDS切り替え時点以降

# その他のロールオーバー手法

- **二重署名法によるZSKロールオーバー**
  - 事前公開法より作業ステップ数が少ないが、ゾーン署名を2回おこなう必要があり、DNSのデータが肥大する
- **事前公開法によるKSKロールオーバー**
  - 親ゾーンとのDSのやりとりが2回必要になる
- **いずれもデメリットが大きいので、あまり使われない**
- **アルゴリズムロールオーバー**
  - 暗号アルゴリズムを変更するときに使われる手法
  - 二重署名法に若干の修正を加えたもの
- **詳しくは以下のインターネットドラフトを参照**
  - DNSSEC Key Timing Considerations
    - http://tools.ietf.org/html/draft-ietf-dnsop-dnssec-key-timing
  - DNSSEC Operational Practices, Version 2
    - http://tools.ietf.org/html/draft-ietf-dnsop-rfc4641bis

# パラメータの設計(1) 暗号アルゴリズム

- **RSASHA256が一般的**
  - ルートゾーン、.jpなど大多数で採用
  - 長いものに巻かれておけばとりあえず問題ない:-)
  - 鍵長はKSK 2048ビット、ZSK 1024ビット
  - RSASHA1は使わない
- **ECDSA、GOST**
  - 楕円曲線暗号
  - RSAよりも小さい鍵長で同等の強度が得られる
    - 署名サイズ、計算量も小さい
    - 将来的にはこっちが主流?
  - 規格化されたのが最近なので、未対応サーバも多い
    - ミスしたときの影響が小さいとも言える

# パラメータの設計(2) 不在証明

- **NSEC vs NSEC3**
  - NSECはゾーン内のすべての情報を取得できてしまう(zone walking)
  - NSEC3はzone walkingはできないが計算量が大きい
- **ゾーン情報を一括取得されて困るかどうかでNSEC/NSEC3のどちらを採用するか判断する**
  - NSEC3でzone　walkingを防いでも、個別に問い合わせがあれば結局答えることには変わりないことに注意
  - DNSの情報は公開が基本であって、秘匿しなければならないようなことはそもそもDNSに載せるべきではない
- **NSEC3を使う場合**
  - iteration回数は大きくしすぎるとパフォーマンスが落ちる(～10回)
  - saltはあまり長くなくてもよいが、定期的に変更する
    - 署名のたびに毎回変更してもよい

# パラメータの設計(3) 鍵の更新間隔

- **KSK: 1年程度**
  - KSKの更新は親ゾーンへのDS登録が必要なので、安全な長さの鍵(2048ビットRSA鍵)を長期間使う
  - ドメインの契約が1年更新なら、その更新時期に合わせてKSKもロールオーバーするとよい
- **ZSK: 1～2ヶ月程度**
  - ZSKの更新はゾーン内部で完結できるので、更新頻度を上げることで安全性の低さ(1024ビット)をカバーする

# パラメータの設計(4) 署名有効期間

- **ゾーン内のレコードの最長TTLより長くする**
  - 署名の有効期間を越えてRRがキャッシュされる危険がある
- **SOA expireよりも長くする**
  - slaveへの転送に失敗し続けると、ゾーンがexpireする前に署名期限を迎えてしまう
- **期間が長すぎるのもよくない**
  - 万が一鍵が漏洩したときに対応できない
- **2週間～2ヶ月ぐらいのところが多いようだ**
- **有効期間の半分が過ぎた程度を目安に再署名して期間を延ばす**
  - 期限切れギリギリの再署名は、キャッシュの状態によっては間に合わないことがある

# 実践編

# DNSSECは煩雑

- **従来のDNS**
  - 年に何回かBINDをアップデート(本来は不要なはず…)
  - 必要に応じてゾーンファイルを更新
- **DNSSEC**
  - さらに加えて…
  - ゾーンファイルへの署名
  - 定期的な鍵の生成、更新(ロールオーバー)
- **非常に煩雑**
  - しかも、手順やコマンドの引数を間違えたときの影響が甚大

# DNSSECはメリットばかりではない

- **DNSSECによって得られるもの**
  - ポイズニングへの耐性
    - とはいえ、DNSSEC validationに対応したキャッシュサーバがまだ微々たるものであることを考えると、現状ではその効果は限定的といえる
    - 将来的に対応キャッシュサーバが増えてくると話は別だが…
  - DNSを利用した新たなアプリケーションの可能性
    - DANEなど、DNSSECで正当性を検証できることを前提に設計されたプロトコル
    - なんでもかんでもDNSに入れるな、という批判もあるけれど…
- **DNSSECによって失われるもの**
  - 手間
    - 通常のDNSの運用に加えて、学習コスト、日々の運用の手間が非常に大きい
  - トラブル対応
    - オペレーションに失敗すると、そのドメインは名前解決不可
    - 事実上、インターネットからドメインが消えるのと同じ
  - DNS amp攻撃の踏み台になる可能性
  - 一部のメール

# qmail 512バイト問題

- **qmailはANYの問い合わせに対して512バイト以上の応答が返ってくるドメインにメールを送れない**
- **DNSSEC署名すると確実に512バイトを越えるため、qmailなサイトからメールが届かなくなる**
  - あくまでqmailのバグであって、DNSSECのせいではない
  - DNSSEC未署名でも、条件を満たせば同様に送信不能
  - パッチあり → http://www.ckdhr.com/ckd/qmail-103.patch
- **DNSSEC化する前にqmailから送られるメールへの対応方針を決めておく必要がある**
  - あるいは、DNSSECを断念する
- http://jprs.jp/tech/notice/2011-03-03-inappropriate-handling-for-long-dns-packet.html

# リスクの低減

- **運用ツールによる自動化**
  - 煩雑で間違えやすい部分は、人間がおこなわず機械的に処理する
  - 鍵ロールオーバーや署名を毎回手作業でおこなうような運用は、むしろミスをおこさない方が奇跡
- **運用の外注**
  - 自前でのDNSSEC運用が覚束ないならば、SIerなどに運用を外部委託するのもひとつの手段
  - とはいえ、現時点ではDNSSEC運用を受けてくれるSIerはあまりなさそう
    - 受けてくれたとして、それが信頼に足るかどうかは未知数

# DNSSEC or not

- **ほんとうにDNSSECは必要ですか?**
  - DNSSECに対応しないのは毒入れのリスクがある
  - DNSSECに対応するのも、また別のリスクを抱えこむことになる
- **リスクを天秤にかけて採用すべきかどうか判断**

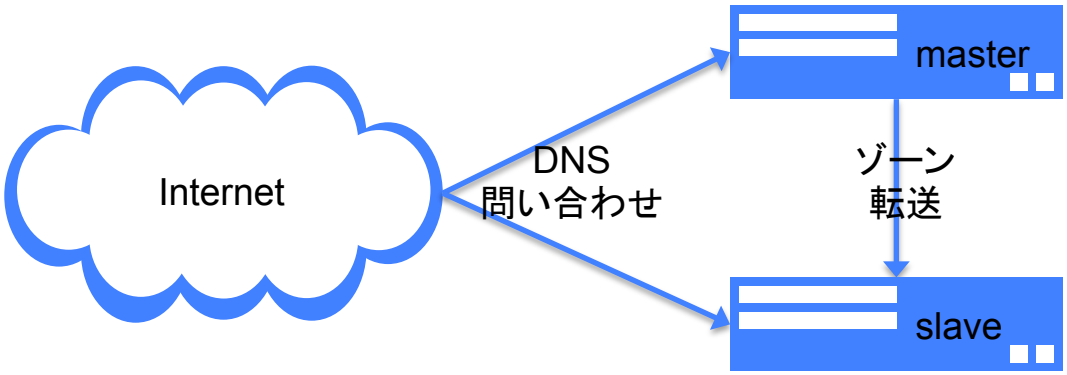
- **覚悟を決めたなら…**
  - OK、DNSSECやりましょう

# サーバの構成検討

- **HTTPSやSSHなどはサーバ上に秘密鍵が必須**
  - リクエストがあるたびその都度暗号化するので
- **DNSSECでは必要ない**
  - 秘密鍵が必要なのは署名するときだけ
  - 問い合わせに対して答えるだけなら不要
  - 万が一サーバに侵入されたとしても、秘密鍵を盗まれなければゾーンが改竄されてもそれを検知できる
  - 外部からアクセスできるサーバには秘密鍵を置かない方がよい
    - 可能ならば、物理的に疎通のない完全なオフライン環境で管理するのが望ましいが、運用の手間は非常に大きくなる

# 従来のDNSサーバ構成

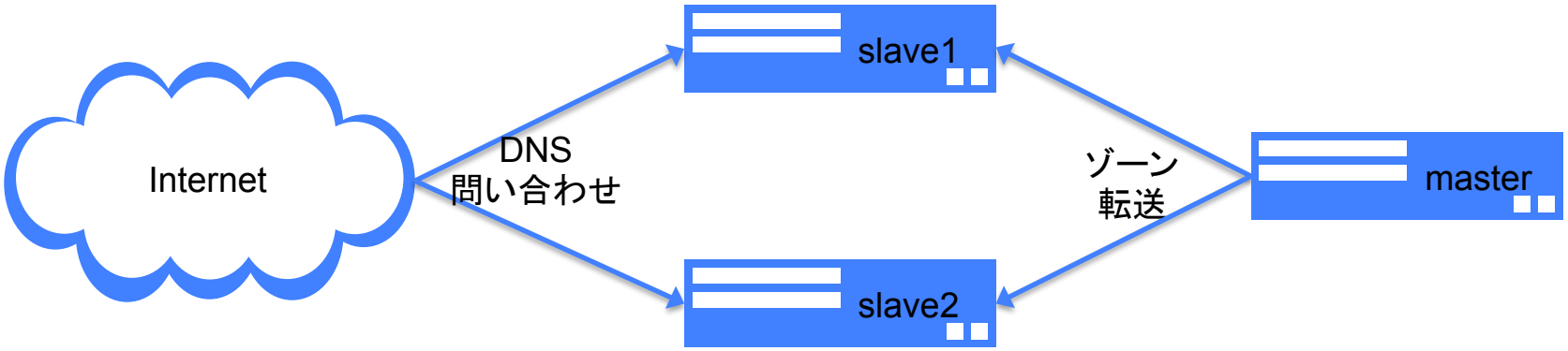
- **master/slaveの2台構成が一般的**
  - masterでゾーンファイルを管理し、slaveに転送
  - masterもslaveもどちらもNSレコードに載せてインターネットからの問い合わせを受ける




```
example.jp.   IN   NS   master.example.jp.
              IN   NS   slave.example.jp.
```

# hidden master構成

- **hidden master = NSレコードに載せないDNSマスタサーバ**
  - 問い合わせを受けるのはslaveの役目
  - masterはゾーンファイルの管理とslaveへのゾーン転送だけをおこなう
    - DNSSECではさらに鍵と署名の管理




```
example.jp.   IN   NS   slave1.example.jp.
              IN   NS   slave2.example.jp.
```

# hidden masterのメリット

- 外に晒すホストに秘密鍵を置かなくてよい
- サーバの役割分担の明確化
  - master: ゾーン、鍵、署名の管理
  - slave: DNS問い合わせ受けつけ
- BINDはDNSSEC管理機能が充実しているけど、セキュリティホール多いよ…
  - 外部と接触しないmasterは機能豊富なBINDを使い、slaveはNSDで安全にサービスする、といった構成が可能
- hidden masterを使う構成が必須というわけではない
  - 従来のmaster - slave構成でももちろん可能

# サーバの負荷

- **DNSSEC処理のための計算量増**
  - とくにNSEC3による不在証明で顕著
- **署名データが付加されることによるトラフィック増**
  - 増えたところでWebサーバなんかと比べたら微々たるものですが…
- **DNSSECをサポートするキャッシュサーバは、DNSSEC検証しない設定になっていてもDOフラグonで問い合わせる**
  - キャッシュサーバ自身が検証しなくても、そのクライアントが検証する可能性があるため、必要な情報を取得しようとする
  - 実際に検証しているかどうかはさておき、DNSSECに対応したバージョンへの置き換えは確実に進んでいる
- **DNSSEC未対応キャッシュサーバからの問い合わせでは、計算量もトラフィックもほとんど変化ない**

# BIND vs NSD

- **DNSSEC関連機能はBINDが圧倒的に充実**
- **NSDは、単体では鍵生成すらできない**
  - 鍵や署名の管理は別途ツールを使うのが前提
  - すでに署名が済んでいれば、あとは十分以上に機能する
- **運用支援ツールはBINDでもNSDでもどちらでも使えることが多い**
  - こういうツールを使うことを前提に考えるならば、BINDの多機能さはむしろ無駄に見える
- **slaveとして使うならそもそも管理機能は不要**
- **慣れと、どんなツールを使って(あるいは使わないで)運用するかを考慮して判断**

# バージョンの確認

- **BINDなら9.[7-9].x (x=最新のもの)**
  - これ以外のものは、DNSSECうんぬん以前にいろいろセキュリティホールがあって使うのは危険
  - RHEL5/CentOS5はbindではなくbind97パッケージを使う
  - RHEL4/CentOS4はRPMパッケージは使わず、最新版ソースから自前でコンパイルする
- **NSDなら3.2.x**
  - 最新版を使っておけばいいかと
- **セカンダリを外部委託している場合、委託先も対応バージョンであることを確認しておく**

# DNSSEC運用支援ツール

- **dnssec-keygenで鍵を作って、dnssec-signzoneを実行して署名や鍵ロールオーバーして、…**
  - DNSSECの学習としては非常に有用
  - どんどんやりましょう
- **が、実運用でそれをやるのは間違いのもと**
  - 自動処理させるべき
- **わけわからん仕事をブラックボックス化するものではない**
  - 裏で何がおこなわれているかはちゃんと把握しておく
- **代表的なものをいくつか紹介しますが、すいません、いずれも自分では使っていないんです…**
  - 自作しちゃったので
  - そんなわけで表面的な紹介に留まりますが、よさそうなものを選んで使ってみてください

# DNSSEC運用支援ツール(1) OpenDNSSEC

- **DNSSEC運用管理ツール**
  - ゾーンファイルへの署名
  - 鍵ロールオーバーの自動化
- **HSM (hardware security module) の使用が前提**
  - 堅牢な鍵管理
  - SoftHSM (ソフトウェア実装したHSM) も提供
- **.se、.uk などいくつかのccTLDでも採用されている**
- http://www.opendnssec.org/

# DNSSEC運用支援ツール(2) DNSSEC-Tools

- **DNSSECの運用全般に役立つツール群**
  - 鍵の管理、ロールオーバー、署名
  - 権威サーバでのDNSSECだけでなく、キャッシュサーバ向けトラストアンカー自動更新ツールや、Nagios、Zabbixの監視プラグイン、ログ解析ツールなども含む
  - 単体で使えるツールが多く、DNSSEC関連のすべてをこれに任せるのではなく、一部だけに利用することも可能
- http://www.dnssec-tools.org/

# DNSSEC運用支援ツール(3) BIND

- **おなじみBINDも、バージョンが新しくなるごとに便利な機能が追加されている**
  - 9.7以降のスマート署名、全自動ゾーン署名
  - 9.9以降のインライン署名
  - ただ、使いやすさという点ではいま一歩の感あり
    - いくつかスクリプトを自作して補う必要あり
  - もっと新しいバージョンではもっと使いやすそうな機能が実装されるのでは、というのが逆に採用をためらわせる
- **ちなみにNSDにはそういう便利な機能は一切ない**
  - あくまでDNSのクエリに答えるだけのサーバであって、管理ツールはほかのものを探してきてくれ、という思想

# ネットワークの確認

- **従来のDNSと比べて、DNSSECでは応答パケットのサイズが大きくなる**
- **大きなUDPパケットをやりとりできることを確認しておく**
  - 512バイト以上のUDPパケットを扱えるか
  - フラグメントしたUDPパケットを組み立てられるか
  - サーバだけでなく、経路上のネットワーク機器も確認
- **DNSSECによりパケットが大きくなっているときであっても、UDPフラグメントはそうめったには発生しない**
  - が、KSKロールオーバー中のDNSKEYの二重署名状態などでは発生することがある
  - 限定的な状況でしか発生しないので、いざひっかかると原因に気がつきにくい
- **もちろんTCPも通ることを確認**

# ドメインレジストラの変更

- **DSレコードの登録・変更は、レジストリに対して直接おこなうのではなく、レジストラを経由しておこなう**
  - NSレコードの登録・変更と同様
- **DSレコードの取り次ぎに対応していないドメインレジストラではDNSSEC化できない**
  - 対応レジストラへの移管が必須
- **が、現状では対応レジストラはわずかしかない**
  - jpの指定事業者一覧には対応状況が記載されている
  - http://jprs.jp/registration/list/
  - jp以外のTLDでもDSを取り次いでくれる国内事業者となるとさらに少ない…

# サーバの時計合わせ

- **RRSIGには有効期間が含まれる**
- **署名の時刻が、それを検証するキャッシュサーバの時刻と大きくズレていると、有効期間外と判断されて検証に失敗する可能性がある**
  - ので、署名するサーバの時計は正しく合わせる
  - もっとも、数分以内のズレならたいてい問題ない
    - ふつうは時計が多少ズレていても問題ないように余裕を持たせた有効期間を設定して署名するので
  - むしろ注意すべきはタイムゾーン
    - 署名の時刻はローカルタイムではなくUTCが使われる
    - ローカルタイムで署名してしまって検証できなかったccTLDというのも…

# SOAシリアルの巻き戻し(1)

- **SOAのシリアル番号はYYYYMMDDnn(年月日+連番2桁)という形式を使っていたところが多いと思いますが…**
- **DNSSECではシリアル番号はunixtime(1970/1/1からの通算秒)が使われることが多い**
  - シリアル番号の管理を機械的に処理しやすくするため
- **2012/11/21を両方の形式で表現してみると…**
  - 2012112101 (YYMMDDnn)
  - 1353423600 (unixtime)
  - YYYYMMDDnnの方が値が大きいので、シリアル番号をunixtimeに変更してもslaveにゾーン転送されない

# SOAシリアルの巻き戻し(2)

- **DNSSEC化する前にシリアル番号を巻き戻す**
  - もちろん、もともとYYMMDDnnでなかった、DNSSEC化してもunixtime形式は使わない、という場合は不要
- **RFC1982 Serial Number Arithmetic**
  - 詳細な説明は割愛します
    - 「DNS　シリアル　巻き戻し」などのキーワードで検索すると詳しく解説されているサイトが見つかります
  - 簡単な手順
    - 現行のシリアル番号の値に 2^31-1 を加えてslaveにゾーン転送
    - 足した結果が 2^32 を越えたらその分減算
    - 目的のシリアル番号に変更して再度ゾーン転送

# DNSSEC対応キャッシュサーバの構築

- **DNSSEC署名しても、それを検証してくれるサーバがないとうまく動いているのか確認ができない**
  - 新たに構築してもいいし、既存キャッシュサーバの設定変更でもよい
- **具体的な構築方法については割愛**
- **テスト時には、上位ゾーンにDSレコードを登録するかわりにキャッシュサーバへのトラストアンカー設定で済ませることができる**
  - 重要なドメインをぶっつけ本番でDNSSEC化したり、テストのために新規ドメインを取得しなくても、既存ドメインにテスト用のサブドメインを作ってそこでテストすればよい

# 練習してみよう

- **本番ドメインをDNSSEC化する前に、テスト用ドメインで練習してみる**
  - 本番ドメインと同じレジストラで同じTLDのドメインを取得することが可能ならば、本番と同じ流れでDS登録も可能
  - が、テストでは手元のキャッシュサーバにトラストアンカーを設定する程度でも十分なことが多い
- **こんな練習をやってみる**
  - 鍵の生成、署名、ロールオーバーをひととおりやってみる
  - わざと手順を間違えてみて、どんな現象が発生するか、どうリカバリーすればいいか考える
- **本番ドメインをDNSSEC化した後も、練習ドメインは残しておくとよい**
  - KSKロールオーバーなんて年に1回しかやらないので、どうせ次回更新時には記憶が薄れている

# 本番ドメインをDNSSEC化してみる

- **練習がうまくいったらいざ本番**
- **DSレコード登録前に最終確認**
  - 署名を開始しても、上位ゾーンにDSを登録するまではDNSSECの検証はおこなわれないのでミスしても大丈夫
  - かわりに手元のキャッシュサーバにトラストアンカーを設定して、署名やロールオーバーなどひととおり試してみる
  - 問題ないという自信があってテストを省略する場合でも、最低限ネガティブキャッシュのTTL以上は待つ
- **ただし、未パッチのqmailによる配送不能問題は、DSを登録しなくても、署名が開始された時点から発生するようになる**
  - 影響を確認しておく
- **問題ないようならば、DSレコードを登録する**
  - これでDNSSEC対応完了です! おめでとう!

# 検証に失敗する…

- **以下の点を確認**
  - ゾーン編集後の再署名を忘れていないか
  - 署名の期限が切れていないか
  - 署名やロールオーバーの手順に間違いはないか
  - KSKと上位ゾーンに登録してあるDSレコードの対応は正しいか
- **検証に成功するキャッシュサーバと失敗するキャッシュサーバが混在している**
  - DNSKEYやRRSIGのキャッシュ状態によって検証結果が変わる
  - ロールオーバー作業時に、十分な待ち時間が経過していないうちに次のステップに進んでしまった、など
  - とくに、DS更新は「レジストラに変更依頼を出した時刻」ではなく、「DSが切り替わったことを自分の目で確認した時刻」を起点に時間をカウントするべき

# DNSSECをやめる手順

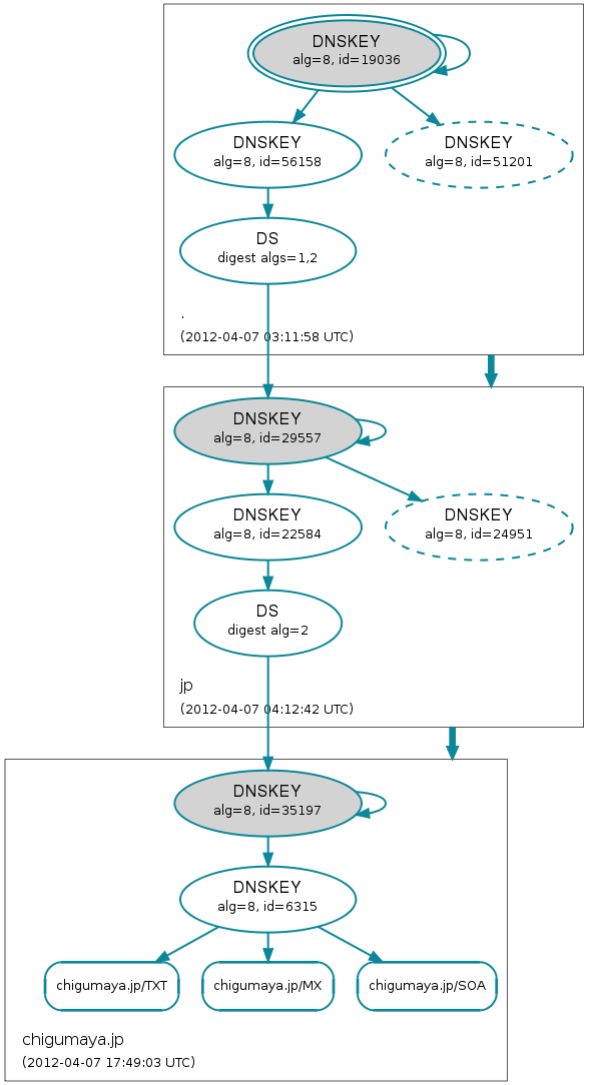
- **すぐにはリカバリーが難しそうならば、いったんDNSSECを止めてしまうのもひとつの選択肢**
  - もちろんその場合はDNSSECによる守りはなくなるので、実際にそうするかどうかはサイトのポリシーによって判断する
- **上位ゾーンからDSレコードを削除する**
- **DSが消えてもキャッシュが残っているうちは署名を続ける**
  - ゾーンを変更したら再署名する
  - 署名有効期限が近づいたら再署名して期間を延ばす
- **DS TTLが過ぎてキャッシュが消えた時点で終了**
  - ゾーンファイルを未署名のものに戻す
- **.jpの場合、DS TTLは24時間**
  - TLDによって異なるので要確認

# 鍵の紛失

- サーバのディスク故障などで秘密鍵にアクセスできなくなってしまった…!
- DS削除でDNSSECを止める
- あるいは、一時的に検証失敗することを承知した上で、正規のロールオーバー手順を踏まずに鍵を切り替える
- 秘密鍵をバックアップしておくと、そのような事態にも対応できる
  - が、鍵のコピーを作るということは、漏洩経路が増えるということも同時に意味することに注意
  - バックアップもオリジナルと同等のセキュリティで管理する

# DNSViz

- **DNSSECの信頼の連鎖を可視化してくれるWebサービス**
  - うまく動いているかの確認に便利
  - http://dnsviz.net/
- **が、ここでDNSSEC検証に失敗と判定されるようなときはすでに手遅れ**
  - キャッシュの状態にもよるが、validationしている世界中のサイトからそのドメインが見えなくなっている
  - ゾーンに署名した後すぐに公開するのではなく、それが問題ないことをテストしてから公開する手順にするとよい



# 署名済みゾーンの事前検証

- ロールオーバー手順が間違っていたり、間違った引数で署名コマンドを実行すると署名を検証できないことがある
  - 間違ってないのに署名コマンドのバグを踏んでトラブルになったTLDの事例もあり…
- 署名したらまずローカルで検証して、問題ないことを確認してから公開するとよい
- ただし、ローカルで動く検証ツールはキャッシュの状態を考慮しないので、状態によって検証の可否が変わるようなチェックはできない
  - 常に正しい手順で作業することを厳守することで正当性を保証する
  - 「正しい手順を心がける」よりも、運用支援ツールで手順をシステム化する

# ローカルで動くゾーンファイル検証ツール

- **YAZVS**
  - http://yazvs.verisignlabs.com/
  - ルート、arpaゾーンは公開前に実際にこのスクリプトで検証している
- **validns**
  - http://www.validns.net/
  - 署名済みゾーンだけでなく、未署名ゾーンの文法チェックも可能
- **donuts**
  - https://www.dnssec-tools.org/wiki/index.php/Donuts
  - プラグイン形式で検証項目をカスタマイズできる
- **Credns**
  - http://www.nlnetlabs.nl/projects/credns/
  - hidden masterから署名済みゾーンを受け取り、検証に成功した場合のみslaveにゾーン転送する

# まとめ

- **権威DNSでのDNSSEC = 鍵と署名の管理**
  - ゾーンを編集したら署名
  - 署名期間が切れる前に再署名
  - 署名鍵の定期ロールオーバー
- **手間が多く、煩雑**
  - しかも失敗したときの影響が大きい
  - すべて手作業で間違いなくやるのは困難
- **手間をかけずに運用できて、人為的な失敗が起きにくくなるように、システム構成全体を見直すべき**
  - 従来のDNSの延長で考えないほうがよい

# 参考資料

- **DNSSEC Operational Practices (RFC4641)**
  - DNSSECを導入するゾーン管理者向けの運用ガイド
  - http://tools.ietf.org/html/rfc4641
- **DNSSEC Operational Practices, Version 2**
  - 上記RFCの改訂版となるべく作業中のインターネットドラフト
  - http://tools.ietf.org/html/draft-ietf-dnsop-rfc4641bis
- **DNSSEC Key Timing Considerations**
  - 鍵ロールオーバーの詳細な手順を解説しているドラフト。RFC化に向けて作業中
  - http://tools.ietf.org/html/draft-ietf-dnsop-dnssec-key-timing