はじめてのDNSSEC

2011年2月5日

NISOC合宿

酔っ払い.JP 運用委員会

本日のテーマ

- 「習うより慣れろ」
  - ➔“とりあえずDNSSECを試す”ができるようになること

本日のテーマ（続き）

本日のスコープ	スコープ外
○(具体的な)設計 ○(具体的な)設定 ○作業手順	×動作解説 ×プロトコル詳細 ×世の中の状況

DNSSEC運用の流れ

事前設計	対象、環境、各種設定値、ライフサイクルを決める
DNSSECの有効化	ゾーン署名、DS登録を行い、検証を有効にする
定常運用	定期的に鍵、署名の更新を行う

事前設計

事前設計

DNSSECの有効化

定常運用

- 対象ドメインの選定
- 利用環境の選定
- DNSSEC固有値の設計
- ライフサイクルの設計

対象ドメインの決定 (1)

- 適用対象
  - 本来は「必要なドメインに」適用するべき。
  - 今回は「失敗しても平気なドメイン」で練習。
- 制限事項
  - 信頼の連鎖の関係上、
    上位ドメインのDNSSEC対応が必要。
  - 同様に「サブドメインで試してみる」も不可。

対象ドメインの決定 (2)

検索タイプ　　検索キーワード

ドメイン名情報　　酔っ払い.JP　　検索

Domain Information: [ドメイン情報]

[ドメイン名]	酔っ払い.JP
[Domain Name]	XN--N8J1C913R6J1B.JP
[登録者名]	酔っ払い協議会
[Registrant]	GUILD OF DRUNKS
[Name Server]	ns1.xn--n8j1c913r6j1b.jp
[登録年月日]	2011/01/13
[有効期限]	2012/01/31
[状態]	Active
[最終更新]	2011/01/13 20:20:53 (JST)

利用環境の選定

- DNSサーバソフトウェア
  - 今回は「BIND9.7系+SmartSigning」で。
- SmartSigning
  - BINDの機能。
    鍵ファイルに時間情報をコメントとして埋め込んでおくと、署名生成コマンド (dnssec-keygen) が適切な鍵を選択してくれる。

DNSSEC固有値の設計(1)

- 決めなくてはいけないこと
  - KSK/ZSK の固有値
    - ✓暗号アルゴリズム
    - ✓鍵長
    - ✓鍵更新方式
  - 不在証明方式

DNSSEC固有値の設計(2)

- KSK/ZSK の固有値
  - ✓ 2011.01現在、RootやJP含め良くみる値にしてみる。

	KSK	ZSK
暗号アルゴリズム	RSASHA256	
鍵長	2048bit	1024bit
鍵更新方式 (*)	二重署名方式	事前公開方式

(*) 詳細 Appendix. I 参照

DNSSEC固有値の設計(3)

- 不在証明方式
  - ✓ 軽いけどzonewalkされそうな NSEC
  - ✓ 重いけどzonewalkされにくい NSEC3
  - ➔ 酔っ払い.JP は A と NS と MX くらいしかないので NSEC を採用。

ライフサイクルの設計(1)

鍵の一生の定義

Create	Publish	Active	Inactive	Delete
鍵の生成。	公開鍵のDNSKEY公開を開始。	秘密鍵による署名生成を開始。	秘密鍵による署名生成を停止。	公開鍵のDNSKEY公開を停止。
DNSKEY: × 署名: ×	DNSKEY: ○ 署名: ×	DNSKEY: ○ 署名: ○	DNSKEY: ○ 署名: ×	DNSKEY: × 署名: ×

- ✓ KSKは更に上位ドメインへのDS登録もある

○: 秘密鍵により署名を生成、もしくは公開鍵をDNSKEYとして公開して*いる*
×: 秘密鍵により署名を生成、もしくは公開鍵をDNSKEYとして公開して*いない*

ライフサイクルの設計(2)

- 決めなくてはいけないこと
  - A) KSKの有効期間
  - B) KSKの二重署名期間
  - C) ZSKの有効期間
  - D) ZSKの事前公開期間
  - E) ゾーン署名の有効期間
  - F) ゾーン署名の更新間隔

ライフサイクルの設計(3)

A) KSKの有効期間
   - まぁ1年くらい？

B) KSKの二重署名期間
   - 上位ドメインにDS登録したりする手間と、確認期間のバッファを積んで1か月くらい？

ライフサイクルの設計(4)

C) ZSKの有効期間
- 数か月に1回変えれば充分な気がするけど・・・cronで実装しやすいから1か月？

D) ZSKの事前公開期間
- DS登録したりする手間と、確認期間のバッファを積んで1か月くらい？

ライフサイクルの設計(5)

E) ゾーン署名の有効期間

- まぁ30日？BINDのデフォルトでもあるし・・・

F) ゾーン署名の更新間隔

- 30日ギリギリで更新するとしくじったら即NGだし、10日毎？

ライフサイクルの設計(6)

- まとめると

No	パラメータ	期間
A	KSKの有効期間	1年
B	KSKの二重署名期間	1か月
C	ZSKの有効期間	1か月
D	ZSKの事前公開期間	1か月
E	ゾーン署名の有効期間	30日
F	ゾーン署名の更新間隔	10日

ライフサイクルの設計(7)

- もっと実装よりに BreakDown【KSK】

No	パラメータ	期間
A	KSKの有効期間	毎年 1/1～12/31
A’	KSKの生成タイミング	毎年12月のどこか
B	KSKの二重署名期間	毎年1/1～1/31

  - KSKは上位ドメイン管理者(他組織)のオペレーションが入るので手動での対応を想定

ライフサイクルの設計(8)

- もっと実装よりに BreakDown【ZSK】

No	パラメータ	期間
C′	ZSKの生成タイミング	毎月20日
D	ZSKの事前公開期間	C′の翌月 1日～末日
C	ZSKの有効期間	Dの翌月 1日～末日
C″	ZSKの削除タイミング	Cの翌月 末日 (*)

  - 自動処理を考えて、cron実装しやすいようにしてみる

(*) C終了後からキャッシュ保持期間経過後に消せるが、手数を減らす為に1ヶ月放置 :p

ライフサイクルの設計(9)

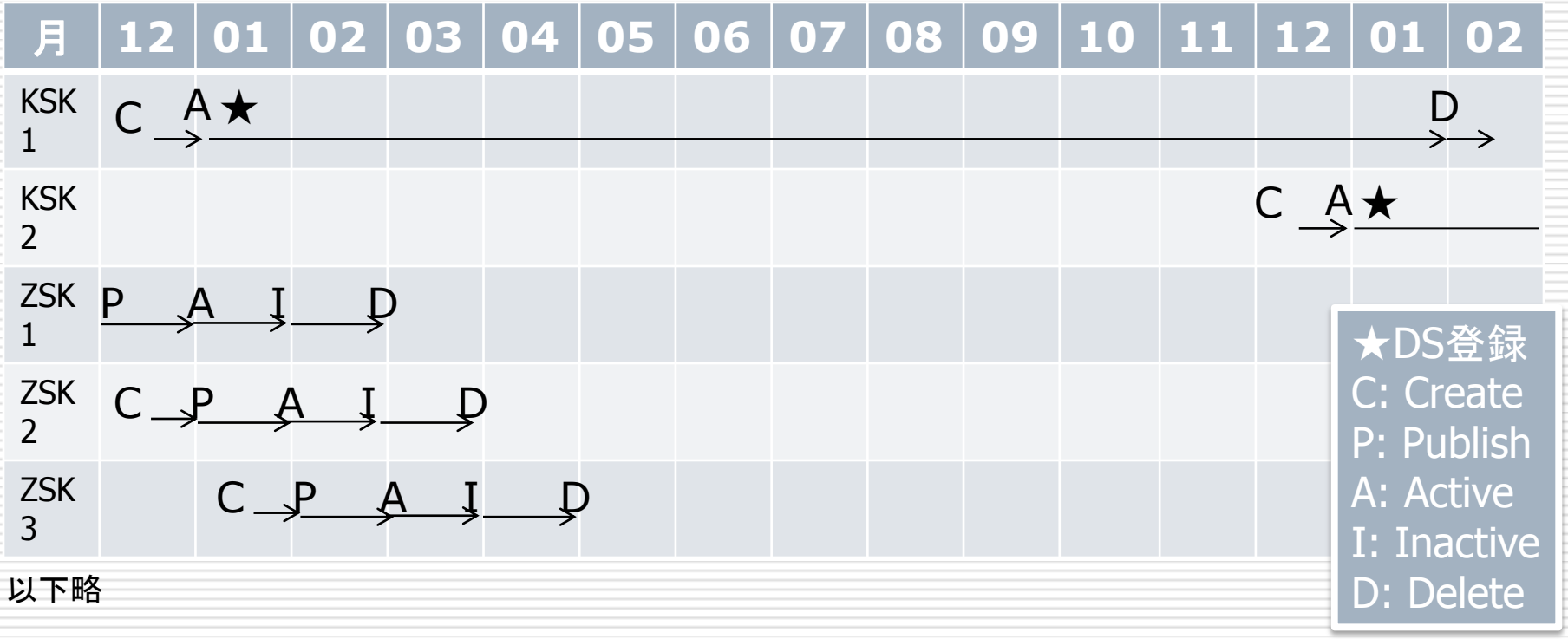
- もっと実装よりに BreakDown【署名】

No	パラメータ	期間
C′	再署名タイミング	毎月1日、11日、21日

  - cronで実装しやすい様に、毎月確実にある日付をいれる。正確に10日ごとじゃないけど気にしない。
  - 1日はZSK更新タイミングの署名も兼ねる。

ライフサイクルの設計(10)

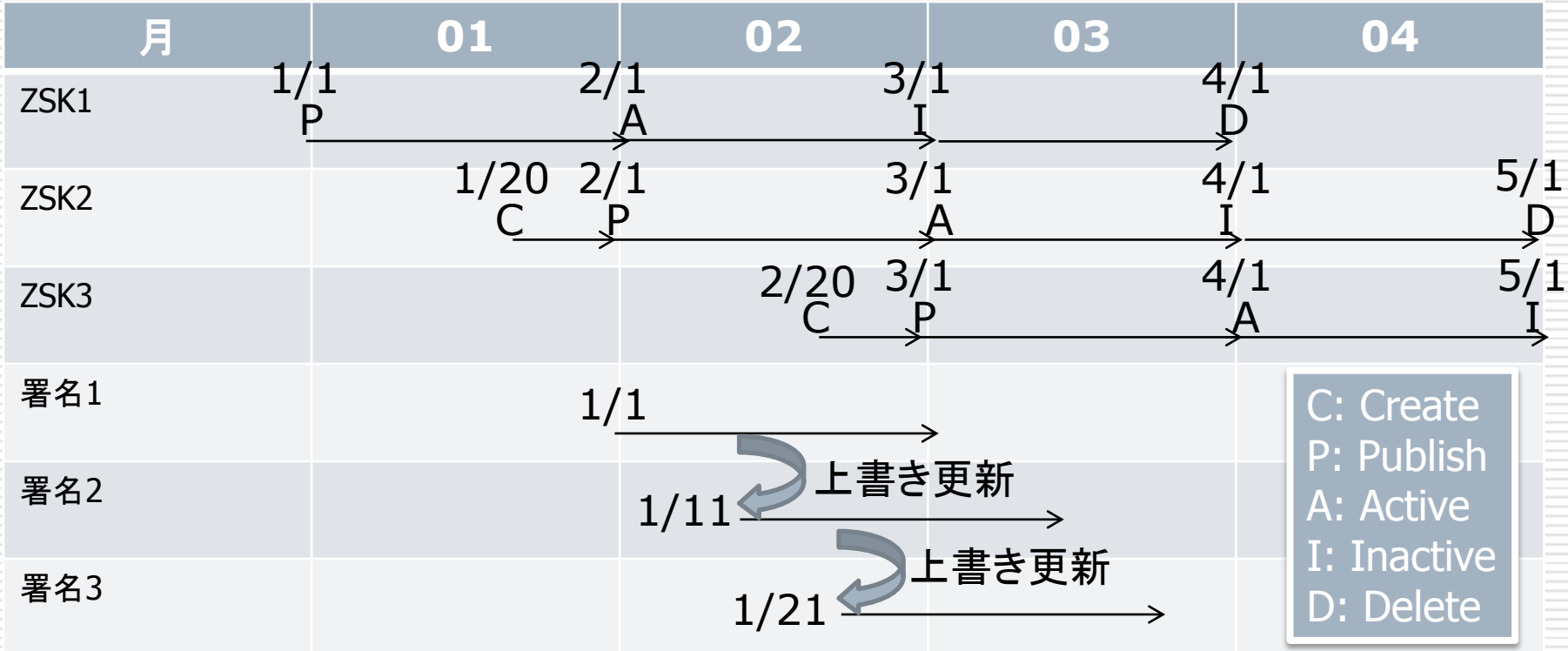
- 図にして書いてみる【KSK/ZSK】



以下略

ライフサイクルの設計(11)

- 図にして書いてみる【ZSK/ゾーン署名】



以下略

DNSSECの有効化

事前設計

DNSSECの有効化

定常運用

鍵の作成

ゾーン署名の実施

上位ドメイン管理者へのDS登録

DNSSECの有効化

- 具体的な例の前に・・・

➔ゾーン署名とDS登録の順序について

  - 必ず、ゾーン署名の後にDS登録をすること！

DSなし、署名あり	DSあり、署名なし
DNSKEY、RRSIGが余計なレコードとして存在するだけ →DNSSECしてない状態と同じ	上位ドメインから参照情報が指定されているのに検証に必要なデータがない →検証失敗、名前解決NG

鍵の作成 (1)

- 今回事例の参考情報
  - 酔っ払い.JPの初回KSK/ZSK/署名生成日
    - ✓ 2011年1月16日
  - JPへの酔っ払い.JPのDS登録日
    - ✓ 2011年1月16日
  - 鍵置き場
    - ✓ /var/named/chroot/etc/pki/dnssec-keys
  - ゾーンファイル置き場
    - ✓ /var/named/chroot/var/named

鍵の作成 (2)

- KSKの作成
  - 初回なのでPublishとActivateは今すぐ。
  - KSKの更新は、上位へのDS登録変更が伴うので、Inactive,Delete(*)は指定しない

```
$ cd /var/named/chroot/etc/pki/dnssec-keys
$ dnssec-keygen -K . ¥
-a RSASHA256 -b 2048 -f KSK ¥
-r /dev/urandom ¥
-P now -A now xn--n8j1c913r6j1b.jp
```

(*)もちろん、上位管理者への更新まで全自動に出来るなら、指定もアリ

鍵の作成 (4)

- ZSKの作成
  - 初回なので、すぐ署名に使う鍵と、次に使う事前公開状態の鍵を作る。
  - ZSKは自動で更新するのでI,Dも決めうち。

	Publish	Active	Inactive	Delete
ZSK1 (すぐ使う)	now	now	2011/02/01 00:00	2011/03/01 00:00
ZSK2 (次に使う)	now	2011/02/01 00:00	2011/03/01 00:00	2011/04/01 00:00

(*)時刻はUTC、dnssec-signzone, dnssec-keygen は基本UTC。

鍵の作成 (5)

- ZSKの作成(ZSK1)

```
$ cd /var/named/chroot/etc/pki/dnssec-keys
$ dnssec-keygen -K . -a RSASHA256 -b 1024 -r /dev/urandom ¥
-P now -A now -I 20110201000000 -D 20110301000000 ¥
xn--n8j1c913r6j1b.jp
```

- ZSKの作成(ZSK2)

```
$ dnssec-keygen -K . -a RSASHA256 -b 1024 -r /dev/urandom ¥
-P now -A 20110201000000 -I 20110301000000 ¥
-D 20110401000000 ¥
xn--n8j1c913r6j1b.jp
```

鍵の作成(6)

- 作成結果

```
/var/named/chroot/etc/pki/dnssec-keys$  ls
Kxn--n8j1c913r6j1b.jp.+008+27963.key
Kxn--n8j1c913r6j1b.jp.+008+27963.private
Kxn--n8j1c913r6j1b.jp.+008+40756.key
Kxn--n8j1c913r6j1b.jp.+008+40756.private
Kxn--n8j1c913r6j1b.jp.+008+44070.key
Kxn--n8j1c913r6j1b.jp.+008+44070.private
```

  - 008はRSASHA256のこと、5桁の数字は鍵タグ、keyは公開鍵、privateは秘密鍵。
  - KSKかZSKかは中味をみないとわからない。。

鍵の作成(7)

- 作成結果
  - ためしにひとつのぞいてみると・・・

```
$ cat Kxn--n8j1c913r6j1b.jp.+008+27963.key
; This is a zone-signing key, keyid 27963, for xn--n8j1c913r6j1b.jp.
; Created: 20110116053939 (Sun Jan 16 14:39:39 2011)
; Publish: 20110116053939 (Sun Jan 16 14:39:39 2011)
; Activate: 20110201000000 (Tue Feb  1 09:00:00 2011)
; Inactive: 20110301000000 (Tue Mar  1 09:00:00 2011)
; Delete: 20110401000000 (Fri Apr  1 09:00:00 2011)
xn--n8j1c913r6j1b.jp. IN DNSKEY 256 3 8
AwEAAdngvlWof38J1IQtIBunhUE8Ybo8
Uyol/lxsMWTaHFYgzrtw9ImjnM10h2JBE
pK4ZiC0m9dHGh+/GfxNaeFprtwnrU73J
ptx/Jh4L
```

ZSKらしい。

SmartSigningで使う日付がコメントとして入っている。日付からするとZSK2らしい。

ゾーン署名の実施(1)

- とりあえず署名してみる

```
$ cd /var/named/chroot/var/named

$ dnssec-signzone ¥
-x ¥       ← DNSKEY RR には ZSKによるRRSIGを生成しない
-S ¥       ← Smart Signing を使用する
-K /var/named/chroot/etc/pki/dnssec-keys/ ¥
-N unixtime ¥    ← 署名実行時に Serial を UNIX time で更新する
xn--n8j1c913r6j1b.jp
```

ゾーン署名の実施(2)

- 今回使わなかったオプション
  - NSEC3を利用する場合には前頁に加えて、
    “-3 [ソルト値] –H [繰返し計算回数]” をつける。

  - 有効期限を手動指定する場合は
    -s 署名の有効期間の開始
    (デフォルトは実行時点の 1h 前)
    -e 署名の有効期間の終了
    (デフォルトは +30日)

ゾーン署名の実施(3)

- 実行結果
  - ゾーンファイル置き場に .signed というファイルができる。

```
$ cd /var/named/chroot/var/named

$ ls –l xn--n8j1c913r6j1b.jp*
-rw-r----- 1 named named  488 Jan 17 15:35 xn--n8j1c913r6j1b.jp
-rw-r--r-- 1 named named 5146 Feb  1 09:00 xn--n8j1c913r6j1b.jp.signed
```

ゾーン署名の実施(4)

- 署名したゾーンファイルの有効化
  - ✓ あとは勇気をだして、named.conf のファイル指定を signed ありにして rndc reload 。

```
zone "xn--n8j1c913r6j1b.jp" {
        type master;
        /* file "xn--n8j1c913r6j1b.jp"; */
        file "xn--n8j1c913r6j1b.jp.signed";
        allow-transfer { localhost; x.x.x.x; };
    };
```

上位ドメイン管理者へのDS登録(1)

- dns-signzone を実行すると、ゾーンファイル置き場にdsset-ドメイン名という名前で、DNSKEYに対応したDSのセットが配置される。

```
$ cd /var/named/chroot/var/named

$ cat dsset-xn--n8j1c913r6j1b.jp.
xn--n8j1c913r6j1b.jp.   IN DS 40756 8 1 872F8B4148E3AB1BFD8BCC45
F9454819CE667B96
xn--n8j1c913r6j1b.jp.   IN DS 40756 8 2 FC9F449CA7769A38A9028BE1E
6220C8CA98E7D34027D5755C526A7C4 0E75FBF5
```

上位ドメイン管理者へのDS登録(2)

- 後は上位ドメイン管理者の様式に従って、登録するだけ、なんだけど・・・
- → 使っているレジストラがDNSSEC対応していない場合は変更しないとダメ。

上位ドメイン管理者へのDS登録(3)

- 注意点
  - DSのcopy&pasteミスが発生すると悲惨なことになるので気を付ける
    - dnssec-dsfromkeyを使うと DNSKEYからDSが作れるので、dig したDNSKEYと突合するのも手
  - 問題が発覚するとしたら、DSが登録された瞬間なのでじーっと見守る

おまけ
酔っ払い.JP イベントカレンダー

有効化

- 1/16昼　KSK,ZSK生成、ゾーン署名実施。
  初回なので、ZSKはすぐ署名に使うもの
  事前公開しておくものを2つ作る
- 1/16 18:15　レジストラ経由でDS登録依頼。
- 1/16 18:30　DS登録され、検証可能に。

定常運用

- 1/20 09:00　ZSK生成(2/1事前公開分)
- 1/21 09:00　署名更新。
- 2/01 09:00　署名更新。自動的に、ZSKも更新される。
- 2/11 09:00　署名更新
- 2/20 09:00　ZSK生成(3/1事前公開分)
- ・・・以下延々と続く

定常運用

事前設計

DNSSECの有効化

定常運用

ゾーンの再署名

ZSKの交換

KSKの交換

定常運用(1)

- ゾーンの再署名
  - dnssec-signzone のラッパーを書いて、cron でまわしてみる
  - UTCとJSTの違いに気をつける。

```
0 9 1,11,21 * * /var/named/bin/SignZone.sh
```

- 通常のゾーン内容変更
  - ゾーン内容変更するときも、再署名するのをわすれない。

定常運用(2)

- ZSKの交換
  - dnssec-keygen のラッパーを 以下同文。
  - 鍵置場に出力すれば、署名のタイミングで dnssec-signzone が条件に合う鍵を使う。
  - 事前公開より前に作っておくと、自動実行に問題が発生してもでリカバリ可能。

```
0 9 20 * * /var/named/bin/CreateZSK.sh
```

  - 今回は翌月1日からの鍵を、20日に作る実装。

定常運用(3)

- 古い鍵の削除
  - dnssec-signzone は鍵置場から条件にある鍵を探すだけなので、このままだとどんどん鍵が増える。
  - 定期的に古い鍵は消してあげる。
- 鍵のバックアップ
  - 鍵データ、特に KSK のデータを無くすと、名前が引けない期間がでちゃって大変なので、きちんとバックアップしておく。

最後に

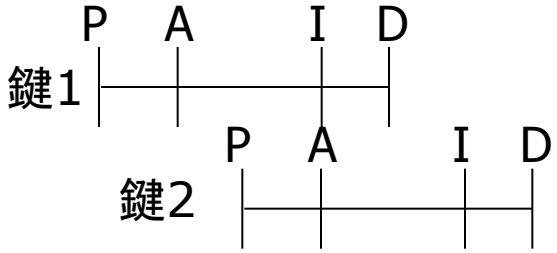
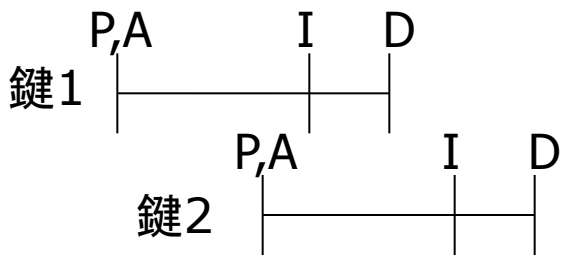
- オススメ
  - 実際に線をひいてみて、ライフサイクルとイベントカレンダーを書く
  - 練習用のドメインで、とにかく鍵更新、署名更新のノウハウを積む
  - 最初はKSK,ZSK,署名の各種更新間隔を短めに設定して、更新機会を増やす

Appendix.I 鍵更新方式 (1)

- DNSの各リソースレコード (RR) にはキャッシュされる時間がある
- → 公開鍵 (DS, DNSKEY) も署名 (RRSIG) も、RRのひとつなのでキャッシュされる
- → キャッシュされた公開鍵(署名)に対応する署名(公開鍵)が参照できる期間が必要

Appendix.I 鍵更新方式 (2)

- 下記いずれかの方式で、キャッシュされたデータで検証可能な組合せが残る様にする

事前公開方式	二重署名方式
署名に用いる前にDS・DNSKEYを公開し、新旧公開鍵が併存する期間(*)をつくる。	署名を新旧双方の秘密鍵で作り、新旧署名が併存する期間(*)をつくる。
	

(*)TTLより長い時間

Appendix.I 鍵更新方式 (3)

- 特徴

事前公開方式	二重署名方式
×PとAを分ける必要がある →運用負荷高 ○署名は1個しかつかない → データ量小	○P,Aが同時でよい →運用負荷低 ×署名が2個ずつつく →データ量大
→ 署名対象が多い (ゾーンデータ) ZSK向き	→ 署名対象が少ない (DNSKEYのみ) KSK向き