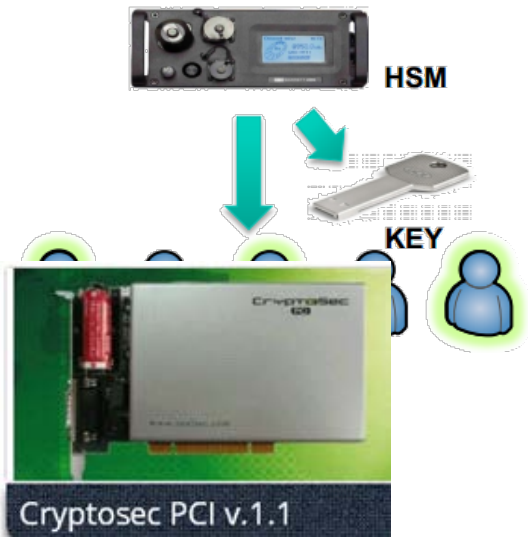
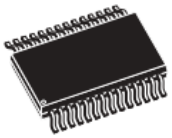
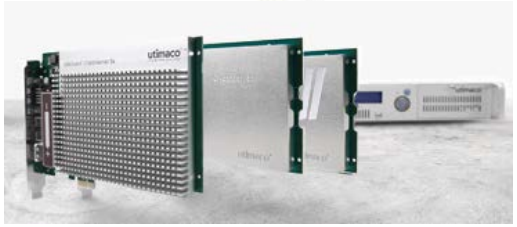
Hardware Security Modules (HSMs) Benefits and Challenges

ICANN 50, London, UK
25 June 2014
richard.lamb@icann.org

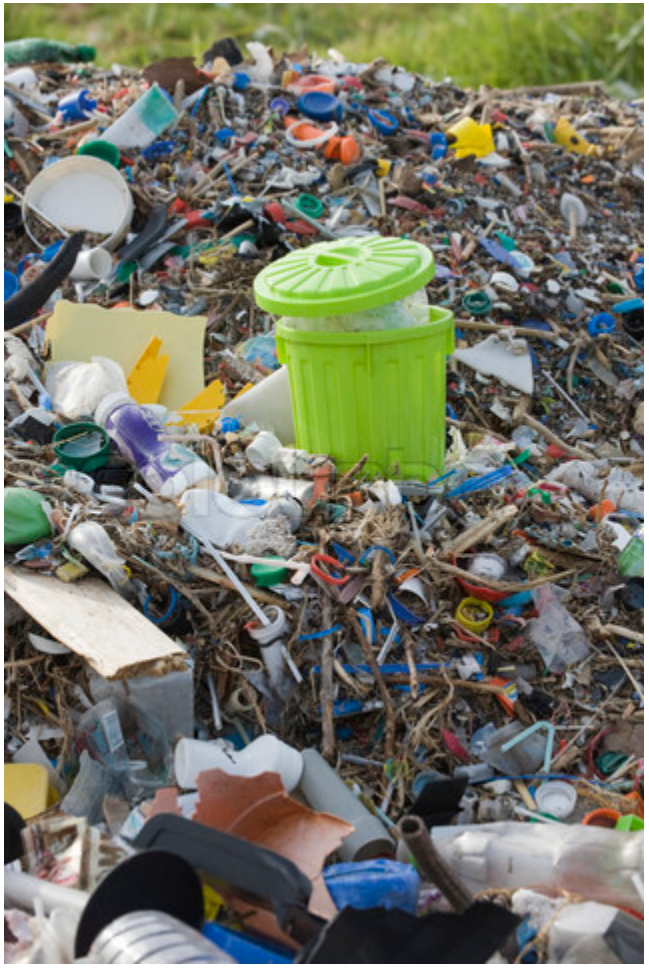
Hardware Security Modules







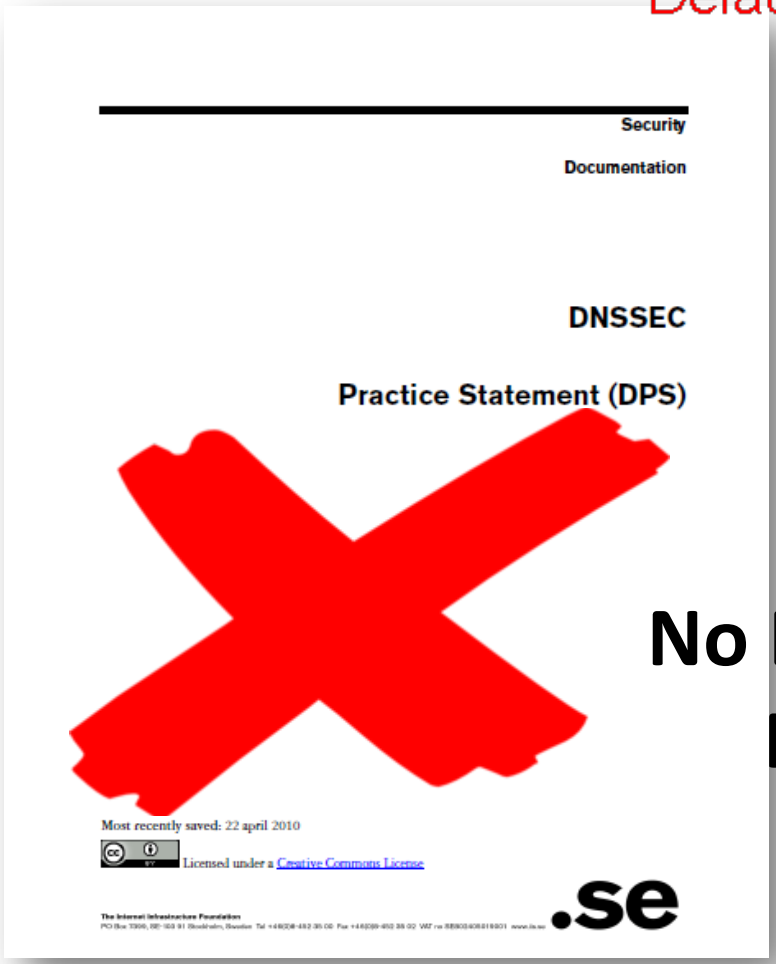
Cool but what are you protecting?

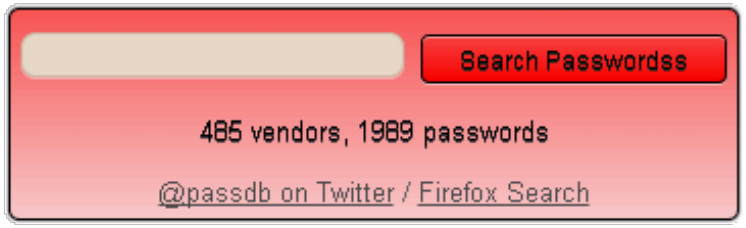


This works fine in many cases

..but this may be the real problem

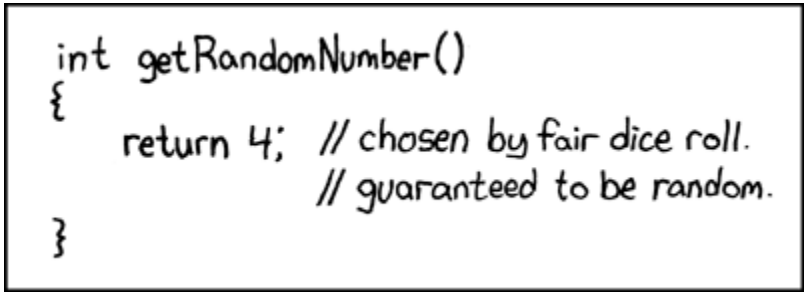
Default Passwords





No Documented Processes

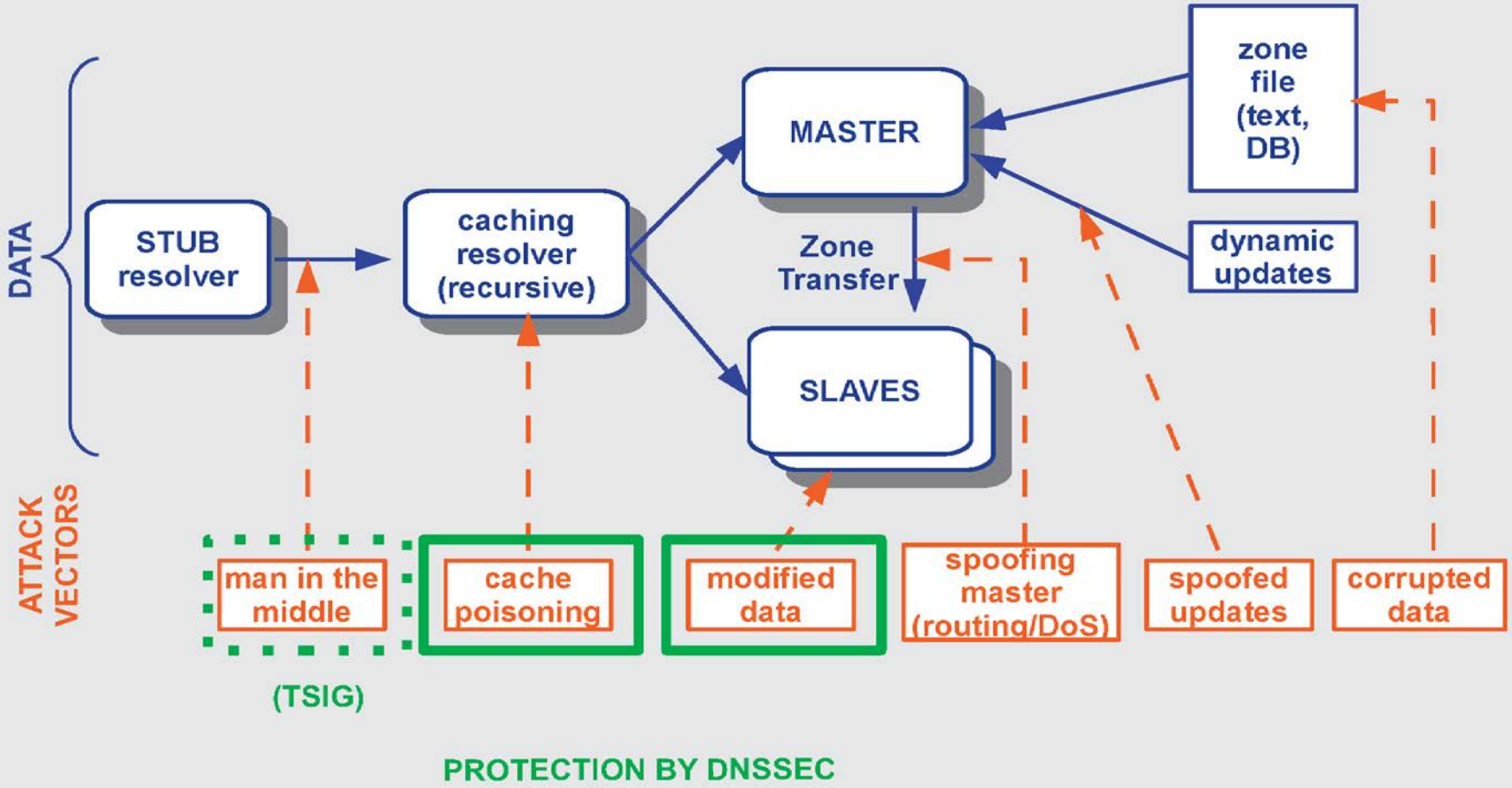
..and sometimes this



Analysis

- What are you protecting?
- Who is your customer?
- What is at risk?
- Set expectations
- Cost

So, what does DNSSEC protect ?



Common API (sort of): PKCS11

- A common interface for HSM and smartcards
  - C_Sign()
  - C_GeneratePair()
- Avoids vendor lock-in – somewhat
  - Also see Key Management Interoperability Protocol (KMIP)
- Vendor Supplied Drivers (mostly Linux, Windows) and some open source

KMIP: http://en.wikipedia.org/wiki/Key_Management_Interoperability_Protocol

Certifications (CYA)

- FIPS 140-2 Level 3
  - Sun SCA6000 (~30000 RSA 1024/sec) ~$10000 (was $1000!!)
  - Thales/Ncipher nshield (~500 RSA 1024/sec) ~$15000
  - Ultimaco
- FIPS 140-2 Level 4
  - AEP Keyper (~1200 RSA 1024/sec) ~$15000
  - IBM 4765 (~1000 RSA 1024/sec) ~$9000
- Recognized by your national certification authority
  - Kryptus (Brazil) ~ $2500
- EAL / Common Criteria
  - >= EAL 4 - Protection Profile for Secure Signature Creation Devices (SSCD) (European standard CWA 14169)

http://www.opendnssec.org/wp-content/uploads/2011/01/A-Review-of-Hardware-Security-Modules-Fall-2010.pdf
http://csrc.nist.gov/groups/STM/cmvp/validation.html
http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm
https://wiki.opendnssec.org/display/DOCREF/HSM+Buyers'+Guide

Smartcards / Tokens

- Smartcards (PKI) (card reader ~$12)
  - AthenaSC IDProtect ~$30 (JP)
  - Feitian ~$5-10 (CN)
  - Aventra ~$11 (FI)
  - CardContact ~$20 (DE)
- TPM
  - Built into many PCs (Messy API)
- Token
  - Aladdin/SafeNet USB e-Token ~$50
- Open source PKCS11 Drivers available
  - OpenSC
- Has RNG
- Slow ~0.5-10 1024 RSA signatures per second

Random Number Generator

- X rand()
- X Netscape: Date+PIDs
- ✓ LavaRand
- ? System Entropy into /dev/random (FBSD=dbrg+entropy/Linux=entropy?)
- ✓ H/W, Quantum Mechanical (laser) $
- ✓ Standards based (FIPS, NIST 800-90 DRBG ;-)
- ✓ Built into CPU chips

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```