Toward Unspoofable Network Identifiers
The Problem

- DNS Spoofing Attacks (e.g., Kaminsky)
- At link (Ethernet) and IP layers, either:
  - Software sets the source address in the packet, or
  - Software can change the “hardware address”
- So: anybody can send a packet with any address
- Bottom Line: no **reliable way** to identify the source of a packet

NOTA BENE: Not clear this is entirely a bad thing. (Why?)
Public-Key Cryptography

- **Principles**
  - Public key can be known to anyone
  - Private key is known only to the principal being authenticated
  - Principal signs message via computation using message and private key
    - Computation produces a sequence of bits: the signature
    - Invariably sign hash of info instead of complete message
  - Authenticator verifies signature:
    - Requires public key, signature, and original message
    - Result of computation indicates authenticity (or not)
Public-Key Cryptography

Issues

- Authenticator must have a way to obtain the signer’s public key
- How can authenticator know it has the right public key?
  - Note: requiring a secure channel to the principal begs the question!

Attempted solution: Certification Hierarchy

- Certificates bind a public key to a digital identifier
- Certification Authority (CA) issues and signs each certificate
- Authenticator must have public key of CA ➔ recurse
- Ultimately there must be some well-known “root” CA public keys

Problem: what identifier to use

- John Smith? 207.110.44.109?
Using PKC to create “Unspoofable” IDs

- DNSSEC
  - RFC’s 4033-4035, 5155
  - Idea: bind public keys to fully-qualified domain names

- Cryptographically-Generated Addresses (CGA)
  - RFC 3972
  - Idea: use hash of public key as identifier
DNSSEC: Using PKC to Secure DNS

- Basic Idea: **Secure zone has a public key**
  - Stored in a DNSKEY record (a new type of RR)
  - Used to sign other records in the zone
  - Signatures are stored in RRSIG records
- Other FQDNs can also have keys (e.g., www.amazon.com)

Simple Example Zone Database without DNSSec:

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.uky.edu</td>
<td></td>
<td>SOA</td>
<td></td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td></td>
<td>NS</td>
<td>ncc.uky.edu</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td></td>
<td>NS</td>
<td>al.cs.uky.edu</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td></td>
<td>A</td>
<td>128.163.1.6</td>
</tr>
<tr>
<td>al.cs.uky.edu</td>
<td></td>
<td>A</td>
<td>128.163.146.100</td>
</tr>
<tr>
<td>neosho.cs.uky.edu</td>
<td></td>
<td>A</td>
<td>128.163.146.22</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td></td>
<td>SOA</td>
<td></td>
</tr>
</tbody>
</table>
**DNSSec: Using PKC to Secure DNS**

Simplified Example Zone Database with DNSSec:

<table>
<thead>
<tr>
<th>Domain</th>
<th>Type</th>
<th>Record</th>
<th>Signature Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>SOA</td>
<td>type covered: SOA [signature]</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>RRSig</td>
<td>type covered: DNSKey [signature]</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>DNSKey</td>
<td>[public key]</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>RRSig</td>
<td>type covered: DNSKey [signature]</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>NS</td>
<td>ncc.uky.edu</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>NS</td>
<td>al.cs.uky.edu</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>RRSig</td>
<td>type covered: NS [signature]</td>
</tr>
<tr>
<td>neosho.cs.uky.edu</td>
<td>IN</td>
<td>A</td>
<td>128.163.146.22</td>
</tr>
<tr>
<td>neosho.cs.uky.edu</td>
<td>IN</td>
<td>RRSig</td>
<td>type covered: A [signature]</td>
</tr>
<tr>
<td>cs.uky.edu</td>
<td>IN</td>
<td>SOA</td>
<td></td>
</tr>
</tbody>
</table>

Signed with cs.uky.edu's private key.
Problem: How to authenticate the statement that no record exists?

Solution: NSEC RR (“Next Secure”)

Each record in a secure zone is followed by an NSEC record, indicating the next FQDN in the zone, and what types of RRs are associated with that FQDN

Based on precise definition of canonical ordering of RRs

NSEC record asserts that “there are no records between this one and the indicated one in the canonical list

<table>
<thead>
<tr>
<th>neosho.cs.uky.edu</th>
<th>IN</th>
<th>A</th>
<th>128.163.146.22</th>
</tr>
</thead>
<tbody>
<tr>
<td>neosho.cs.uky.edu</td>
<td>IN</td>
<td>NSEC</td>
<td>nobby.cs.uky.edu</td>
</tr>
</tbody>
</table>
DNSSec: The Whole Picture

Simplified Example Zone Database including NSEC:

<table>
<thead>
<tr>
<th>CS.UKY.EDU</th>
<th>IN</th>
<th>SOA</th>
<th>CS.UKY.EDU</th>
<th>IN</th>
<th>DNSKey [public key]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS.UKY.EDU</td>
<td>IN</td>
<td>RRSig</td>
<td>TYPE COVERED: SOA [signature]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS.UKY.EDU</td>
<td>IN</td>
<td>NS</td>
<td>NCC.UKY.EDU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS.UKY.EDU</td>
<td>IN</td>
<td>NS</td>
<td>AL.CS.UKY.EDU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS.UKY.EDU</td>
<td>IN</td>
<td>RRSig</td>
<td>TYPE COVERED: NS [signature]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS.UKY.EDU</td>
<td>IN</td>
<td>NSEC</td>
<td>NEOSHO.CS.UKY.EDU A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS.UKY.EDU</td>
<td>IN</td>
<td>RRSig</td>
<td>TYPE COVERED: NSEC [signature]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEOSHO.CS.UKY.EDU</td>
<td>IN</td>
<td>A</td>
<td>128.163.146.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEOSHO.CS.UKY.EDU</td>
<td>IN</td>
<td>RRSig</td>
<td>TYPE COVERED: A [signature]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEOSHO.CS.UKY.EDU</td>
<td>IN</td>
<td>NSEC</td>
<td>CS.UKY.EDU SOA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEOSHO.CS.UKY.EDU</td>
<td>IN</td>
<td>RRSig</td>
<td>TYPE COVERED: NSEC [signature]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS.UKY.EDU</td>
<td>IN</td>
<td>SOA</td>
<td>CS.UKY.EDU SOA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Canonical Order

eample
a.example
yljkjlk.a.example
Z.a.example
zABC.a.EXAMPLE
z.example
\001.z.example
*.z.example
\200.z.example
DNSSEC Issues

- **All-or-nothing**: spec (RFC 4034 & 4035) says entire zone must be secure or not
- ~Quadruples the size of the database
  - RRSig RR (x 2) plus NSEC RR for each existing RRset
  - New RR types are large (containing digital signatures)
- **NSEC allows enumeration of all names in a zone**
  - This is considered a bad thing (aid to intruders)
  - NSEC3 RR introduced as a fix
    - NSEC3 uses a hash chain to provide authenticated denial of existence
- **Parent zones must be secure**
  - MUST be sure you are talking to an authoritative nameserver for the secure zone
  - Root zone has well-known public key
Cryptographically-Generated Addresses (CGA)

- Well-known idea: if your identifier space is large enough and unstructured, you can use a hash of the public key as the identifier: self-certifying identifiers!
- Would like to use this to bind public keys to addresses
- Note: security of self-certifying IDs rests on properties of hash functions
  - Viz: infeasible to find a collision
- Problem: addresses are neither unstructured nor large enough
Cryptographically Generated Addresses

- IPv6 only
- Concept: use (part of) the hash of the public key for the interface identifier of the IPv6 address

<table>
<thead>
<tr>
<th>Routing Prefix</th>
<th>interface identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bits</td>
<td>64 bits</td>
</tr>
</tbody>
</table>

IPv6 Address

- Problem: 64 bits is not enough to be secure
Cryptographically Generated Addresses

- Solution: Require that some number of the high-order bits of the hash of the public key be 0
  - How many is determined by the security parameter, which is part of the address

<table>
<thead>
<tr>
<th>Routing Prefix</th>
<th>interface identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bits</td>
<td>3 bits</td>
</tr>
</tbody>
</table>

IPv6 Address

- Computation: choose a random 128-bit string; compute SHA-1 hash of it. If the high-order (7 x sec value) bits are zero, do another hash to create the interface ID
Self-certifying IPv6 Addresses

Advantages:
- Completely distributed address assignment
- No PKI required

Disadvantages:
- To authenticate an IPv6 datagram, sender must include:
  - Public key
  - Signature on (hash of) the contents of the datagram
- Heavy computational burden in creation of interface ID
  - Work grows exponentially with security parameter
  - Find a hash value with 16, 32, or more leading zeros