Primary-Secondary-Resolvers Membership Proof Systems and their Applications to DNSSEC

Weizmann Institute

Sharon Goldberg, Moni Naor, Dimitris Papadopoulos,
Leonid Reyzin, Sachin Vasant, Asaf Ziv

The (non) membership problem

- Database R of n elements from universe $\boldsymbol{U}$
  - With object $x \in R$ associated information y
- Want to allow **lookups** in R such that
  - If $x \in R$ then answer is '**yes**' and associated y retrieved
  - If $x \notin R$ then answer is '**no**'
- Don't want to leak more information than this!
- Entity providing answer: **not** trusted wrt to correctness.

Learns if x is in R

Primary ⟹ **Secondary** ⟺ **Resolver**

Trusted, **Offline**

Not trusted, **Online**

Has $x \in \boldsymbol{U}$
knows primary' public key

Motivation: Secure DNS Lookups

Example.com: 172.16.254.1

- DNS: Domain Name Server
  - Allows the translation of names to IP Addresses
  - Plain DNS does **not** guarantee **authenticity** to users
- DNSSEC: Security extension of DNS
  - Retrieved records are authenticated (signed)
  - What about non-exiting records? **Denial of existence**
  - Current methods leak information about the set
  - Allow `zone enumeration’

Listing all names in a domain

- Want to improve DNSSEC

How NSEC Works (Roughly)

- The **primary** signs all existing records
  - plus link to the next record in sorted order
  - Gives all signatures to secondary
  - Public key: signing key
- Given query x
  - If $x \in R$ then secondary gives signature on record
  - If $x \notin R$ then **proof of non existence** is:

signed pair (x_1, x_2) such that $x_1 < x < x_2$

After a while: learn all of R

- Even random queries

Primary ⟹ **Secondary** ⟺ **Resolver**

Trusted, **Offline**

Not trusted, **Online**

Has $x \in U$ knows primary' public key

Is Zone Enumeration a Real Problem?

Much debate in the networking world:After all this is public information?

- There is a difference between **willing to answer questions** and **revealing everything you know**
- Enumerating hostnames creates a toehold for more complex attacks
- Legal reasons to protect host names (e.g. EU Data Protection laws)
- IETF rewrote the DNSSEC standard to `deal' with this issue in 2008

How NSEC3 Works (Roughly)

May also add salt

- Instead of storing x itself: store h(x)
  - h **is some one-way/random oracle function**
- The problem is now similar to the case where one is given oracle access to the membership function
  - At best: this is an **obfuscated membership program** and allows the adversary ``unlimited" queries
- Bernstein's NSEC3 walker

What Do We Have to Say

- **Model** the problem
  - Primary-Secondary-Resolvers Membership Proof Systems

Completeness, Soundness &
Privacy (Zero-Knowledge)

- **Explain** why current attempts have all failed
  - Show that the **secondary** must be performing **online public-key authentication**
  - Can convert to **signatures** in some circumstances
- **Suggest** various constructions to PSRs
  - Based on RSA plus random oracles
  - Based on VRFs and VUFs
  - Based on HIBEs

NSEC5

How Our NSEC5 Works (Roughly)

- Instead of storing x itself: store

$$F(x)=h_2(RSA^{-1}(h_1(x)))$$

where h_1 and h_2 are random oracles

- Unlike $h(x)$ in NSEC3: not everybody can compute it.
- Equip the **secondary** with the RSA secret key
- To prove that $F(x)=z$:
  - secondary sends $S(x)=RSA^{-1}(h_1(x))$
- **Resolver** needs to know public RSA key
  - One additional RSA computation

How NSEC5 Works (Roughly)

Primary preparation

- Choose Signing key **plus** RSA key **(N,e)** and hash functions h_1: $U \rightarrow [N]$ and h_2: $[N] \rightarrow \{0,1\}^\wedge$

Random oracles

Denote $S(x)=RSA^{-1}(h_1(x))$ and $F(x)=h_2(S(x))$

Plays the role of $h(x)$ in NSEC3

- For every $x_i \in R$ compute $y_i=F(x_i)$
- Sign them in **pairs** by **lexicographical** order: $Sign(y_i, y_{i+1})$
- For every $x_i \in R$ also sign their values: $Sign(x_i, v_i)$

Secondary's Public key PK_S **=(N,e)**

Secondary's secret key SK_S **=d** and

- Set R and $Sign(x_i, v_i)$
- For all pairs $Sign(y_i, y_{i+1})$

$\Bigg\} \; I_S$

NSEC5 RSA Construction

Denote $S(x)=RSA^{-1}(h_1(x))$ and $F(x)=h_2(S(x))$

- For every $x_i \in R$ compute $y_i=F(x_i)$
- Sign them in **pairs** by **lexicographical** order: $Sign(y_i, y_{i+1})$
- For every $x_i \in R$ also sign their values: $Sign(x_i, v_i)$

Secondary

- Given a query $x \in R$, the secondary returns $Sign(x_i, v_i)$
- Given query $x \notin R$, the secondary returns:

  $Sign(y_i, y_{i+1})$ **and** $S(x)$ such that $y_i < F(x) < y_{i+1}$

A **Resolver** verifies query x by checking that:

- $y_i < h_2(S(x)) = F(x) < y_{i+1}$
- $RSA(S(x))=h_1(x)$

NSEC5 RSA Performance

Performance comparable to NSEC3

Primary: Signature on **pairs** $Sign(y_i, y_{i+1})$

Signature on values: $Sign(x_i, v_i)$

For every $x_i \in R$ compute $y_i=F(x_i)$

From lower bound: must work as hard as signing!

Secondary

- For query $x \notin R$: secondary **computes** $y=F(x)$ and returns: $Sign(y_i, y_{i+1})$ **and** $S(x)$

A **Resolver** verifies query x by checking that:

- $y_i < h_2(S(x)) = F(x) < y_{i+1}$
- $RSA(S(x)) = h_1(x)$

Based on

- NSEC5: Provably Preventing DNSSEC Zone Enumeration Sharon Goldberg, Moni Naor, Dimitris Papadopoulos, Leonid Reyzin, Sachin Vasant, Asaf Ziv

  Cryptology ePrint Archive: Report 2014/582

- PSR Membership Proof Systems, Moni Naor and Asaf Ziv

ITCS 2015 at Weizmann Institute

- The **6th Innovations in Theoretical Computer Science (ITCS)** conference, will be held at the Weizmann Institute of Science, Israel
  January 11-13, 2015

- **Deadline: Aug 8th 2014**
- Program Chair: Tim Roughgarden

http://www.wisdom.weizmann.ac.il/~naor/itcs2015_main.html

- Sponsored by ACM SIGACT