DNSSEC goals and non-goals

+ Authenticity of positive replies
+ Authenticity of negative replies
+ Integrity
  - Confidentiality
  - Availability

Negative answers

- Also don’t want attackers to spoof non-existence
  - Gratuitous denial of service, force fallback, etc.
- But don’t want to sign “x does not exist” for all x
- Solution 1, NSEC: “there is no name between acacia and baobab”

Preventing zone enumeration

- Many domains would not like people enumerating all their entries
- DNS is public, but “not that public”
- Unfortunately NSEC makes this trivial
- Compromise: NSEC3 uses password-like salt and repeated hash, allows opt-out

DANE: linking TLS to DNSSEC

- “DNS-based Authentication of Named Entities”
- DNS contains hash of TLS cert, don’t need CAs
- How is DNSSEC’s tree of certs better than TLS’s?
Signing the root

- Political problem: many already distrust US-centered nature of DNS infrastructure
- Practical problem: must be very secure with no single point of failure
- Finally accomplished in 2010
  - Solution involves ‘key ceremonies’, international committees, smart cards, safe deposit boxes, etc.

Deployment

- Standard deployment problem: all cost and no benefit to being first mover
- Servers working on it, mostly top-down
- Clients: still less than 20%
- Will be probably common: insecure connection to secure resolver

Outline

- DNSSEC, cont’d
- SSH
- Announcements intermission
- More crypto protocols
- More causes of crypto failure

Short history of SSH

- Started out as freeware by Tatu Ylönen in 1995
- Original version commercialized
- Fully open-source OpenSSH from OpenBSD
- Protocol redesigned and standardized for “SSH 2”

OpenSSH t-shirt

SSH host keys

- Every SSH server has a public/private keypair
- Ideally, never changes once SSH is installed
- Early generation is a classic entropy problem
  - Especially embedded systems, VMs
**Authentication methods**

- Password, encrypted over channel
- `.shosts`: like `.rhosts`, but using client host key
- User-specific keypair
  - Public half on server, private on client
- Plugins for Kerberos, PAM modules, etc.

**Old crypto vulnerabilities**

- 1.x had only CRC for integrity
  - Worst case: when used with RC4
- Injection attacks still possible with CBC
  - CRC compensation attack
- For least-insecure 1.x-compatibility, attack detector
  - Alas, detector had integer overflow worse than original attack

**Newer crypto vulnerabilities**

- IV chaining: IV based on last message ciphertext
  - Allows chosen plaintext attacks
  - Better proposal: separate, random IVs
- Some tricky attacks still left
  - Send byte-by-byte, watch for errors
  - Of arguable exploitability due to abort
- Now migrating to CTR mode

**SSH over SSH**

- SSH to machine 1, from there to machine 2
  - Common in these days of NATs
- Better: have machine 1 forward an encrypted connection (cf. HW1)
  1. No need to trust 1 for secrecy
  2. Timing attacks against password typing

**SSH (non-)PKI**

- When you connect to a host freshly, a mild note
- When the host key has changed, a large warning

```
WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now
(man-in-the-middle attack)!

It is also possible that a host key has just been changed.
```

**Outline**

- DNSSEC, cont’d
- SSH
- Announcements intermission
- More crypto protocols
- More causes of crypto failure
Upcoming assignments

- Hands-on assignment 2 is due Friday
- For best results, don’t put off until last minute

Outline

DNSSEC, cont’d
SSH
Announcements intermission
More crypto protocols
More causes of crypto failure

Abstract protocols

- Outline of what information is communicated in messages
- Omit most details of encoding, naming, sizes, choice of ciphers, etc.
- Describes honest operation
  - But must be secure against adversarial participants
- Seemingly simple, but many subtle problems

Protocol notation

\[ A \rightarrow B : N_B, \{ T_0, B, N_B \}_K_B \]
\[ A \rightarrow B : \text{message sent from Alice intended for Bob} \]
\[ B \text{ (after :): Bob’s name} \]
\[ \{ \ldots \}_K : \text{encryption with key } K \]

Needham-Schroeder

Mutual authentication via nonce exchange, assuming public keys (core):
\[ A \rightarrow B : \{ N_A, A \}_E_B \]
\[ B \rightarrow A : \{ N_A, N_B \}_E_A \]
\[ A \rightarrow B : \{ N_B \}_E_B \]

Needham-Schroeder MITM

\[ A \rightarrow C : \{ N_A, A \}_E_C \]
\[ C \rightarrow B : \{ N_A, A \}_E_B \]
\[ B \rightarrow C : \{ N_A, N_B \}_E_A \]
\[ C \rightarrow A : \{ N_A, N_B \}_E_A \]
\[ A \rightarrow C : \{ N_B \}_E_C \]
\[ C \rightarrow B : \{ N_B \}_E_B \]
Certificates, Denning-Sacco

A certificate signed by a trusted third-party $S$ binds an identity to a public key

$$C_A = \text{Sign}_S(A, K_A)$$

Suppose we want to use $S$ in establishing a session key $K_{AB}$:

$A \rightarrow S : A, B$

$S \rightarrow A : C_A, C_B$

$A \rightarrow B : C_A, C_B, \{\text{Sign}_A(K_{AB})\}_{K_B}$

Attack against Denning-Sacco

$A \rightarrow S : A, B$

$S \rightarrow A : C_A, C_B$

$A \rightarrow B : C_A, C_B, \{\text{Sign}_A(K_{AB})\}_{K_B}$

$B \rightarrow S : B, C$

$S \rightarrow B : C_B, C_C$

$B \rightarrow C : C_A, C_C, \{\text{Sign}_A(K_{AB})\}_{K_C}$

By re-encrypting the signed key, Bob can pretend to be Alice to Charlie

Envelopes analogy

Encrypt then sign, or vice-versa?

On paper, we usually sign inside an envelope, not outside. Two reasons:

- Attacker gets letter, puts in his own envelope (c.f. attack against X.509)
- Signer claims “didn’t know what was in the envelope” (failure of non-repudiation)

Design robustness principles

- Use timestamps or nonces for freshness
- Be explicit about the context
- Don’t trust the secrecy of others’ secrets
- Whenever you sign or decrypt, beware of being an oracle
- Distinguish runs of a protocol

Implementation principles

- Ensure unique message types and parsing
- Design for ciphers and key sizes to change
- Limit information in outbound error messages
- Be careful with out-of-order messages

Outline

DNSSEC, cont’d

SSH

Announcements intermission

More crypto protocols

More causes of crypto failure
Random numbers and entropy

Cryptographic RNGs use cipher-like techniques to provide indistinguishability.
But rely on truly random seeding to stop brute force.
- Extreme case: no entropy $\rightarrow$ always same "randomness"
- Modern best practice: seed pool with 256 bits of entropy
  - Suitable for security levels up to $2^{256}$

Netscape RNG failure

Early versions of Netscape SSL (1994-1995) seeded with:
- Time of day
- Process ID
- Parent process ID
- Best case entropy only 64 bits
  - (Not out of step with using 40-bit encryption)
- But worse because many bits guessable

Debian/OpenSSL RNG failure (1)

OpenSSL has pretty good scheme using /dev/urandom.
Also mixed in some uninitialized variable values
- "Extra variation can't hurt"
From modern perspective, this was the original sin
  - Remember undefined behavior discussion?
- But had no immediate ill effects

Debian/OpenSSL RNG failure (2)

Debian maintainer commented out some lines to fix a Valgrind warning
- "Potential use of uninitialized value"
Accidentally disabled most entropy (all but 16 bits)
Brief mailing list discussion didn't lead to understanding
Broken library used for \(~2\) years before discovery

Detected RSA/DSA collisions

2012: around 1% of the SSL keys on the public net are breakable
- Some sites share complete keypairs
- RSA keys with one prime in common (detected by large-scale GCD)
One likely culprit: insufficient entropy in key generation
  - Embedded devices, Linux /dev/urandom vs. /dev/random
- DSA signature algorithm also very vulnerable

New factoring problem (CCS'17)

An Infineon RSA library used primes of the form $p = k \cdot M + (65537^a \mod M)$
- Smaller problems: fingerprintable, less entropy
- Major problem: can factor with a variant of Coppersmith's algorithm
  - E.g., 3 CPU months for a 1024-bit key
Side-channel attacks
- Timing analysis:
  - Number of 1 bits in modular exponentiation
  - Unpadding, MAC checking, error handling
  - Probe cache state of AES table entries
- Power analysis
  - Especially useful against smartcards
- Fault injection
- Data non-erasure
  - Hard disks, “cold boot” on RAM

WEP “privacy”
- First WiFi encryption standard: Wired Equivalent Privacy (WEP)
- F&S: designed by a committee that contained no cryptographers
- Problem 1: note “privacy”: what about integrity?
  - Nope: stream cipher + CRC = easy bit flipping

WEP shared key
- Single key known by all parties on network
- Easy to compromise
- Hard to change
- Also often disabled by default
- Example: a previous employer

WEP key size and IV size
- Original sizes: 40-bit shared key (export restrictions) plus 24-bit IV = 64-bit RC4 key
  - Both too small
- 128-bit upgrade kept 24-bit IV
  - Vague about how to choose IVs
  - Least bad: sequential, collision takes hours
  - Worse: random or everyone starts at zero

WEP RC4 related key attacks
- Only true crypto weakness
- RC4 “key schedule” vulnerable when:
  - RC4 keys very similar (e.g., same key, similar IV)
  - First stream bytes used
- Not a practical problem for other RC4 users like SSL
  - Key from a hash, skip first output bytes

New problem with WPA (CCS’17)
- Session key set up in a 4-message handshake
- Key reinstallation attack: replay #3
  - Causes most implementations to reset nonce and replay counter
  - In turn allowing many other attacks
  - One especially bad case: reset key to 0
- Protocol state machine behavior poorly described in spec
  - Outside the scope of previous security proofs
## Trustworthiness of primitives

- Classic worry: DES S-boxes
- Obviously in trouble if cipher chosen by your adversary
- In a public spec, most worrying are unexplained elements
- Best practice: choose constants from well-known math, like digits of $\pi$

## Dual_EC_DRBG (1)

- Pseudorandom generator in NIST standard, based on elliptic curve
- Looks like provable (slow enough!) but strangely no proof
- Specification includes long unexplained constants
- Academic researchers find:
  - Some EC parts look good
  - But outputs are statistically distinguishable

## Dual_EC_DRBG (2)

- Found 2007: special choice of constants allows prediction attacks
  - Big red flag for paranoid academics
- Significant adoption in products sold to US govt. FIPS-140 standards
  - Semi-plausible rationale from RSA (EMC)
- NSA scenario basically confirmed by Snowden leaks
  - NIST and RSA immediately recommend withdrawal