CSci 5271 Introduction to Computer Security More crypto protocols and failures

Stephen McCamant University of Minnesota, Computer Science & Engineering

Outline

More crypto protocols

Announcements intermission

More causes of crypto failure

Abstract protocolsProtocol notation Outline of what information is communicated in messagesOmit most details of encoding, naming, sizes, choice of ciphers, etc.Describes honest operationBut must be secure against adversarial participantsSeemingly simple, but many subtle problems $A \to B : N_B, \{T_0, B, N_B\}_{K_B}$ $A \to B : N_B, \{T_0, B, N_B\}_{K_B}$ $B = M_B = M_B = M_B + M_B$

Needham-Schroeder

Mutual authentication via nonce exchange, assuming public keys (core):

 $\begin{array}{l} A \rightarrow B: \ \{N_A,A\}_{E_B} \\ B \rightarrow A: \ \{N_A,N_B\}_{E_A} \\ A \rightarrow B: \ \{N_B\}_{E_B} \end{array}$

Needham-Schroeder MITM

 $\begin{array}{l} A \rightarrow C: \ \{N_A,A\}_{E_C} \\ C \rightarrow B: \ \{N_A,A\}_{E_B} \\ B \rightarrow C: \ \{N_A,N_B\}_{E_A} \\ C \rightarrow A: \ \{N_A,N_B\}_{E_A} \\ A \rightarrow C: \ \{N_B\}_{E_C} \\ C \rightarrow B: \ \{N_B\}_{E_B} \end{array}$

Certificates, Denning-Sacco
 A certificate signed by a trusted third-party S binds an identity to a public key C_A = Sign_S(A, K_A)
🖲 Suppose we want to use S in establishing a session
$\begin{array}{rll} & A \rightarrow S: & A, B \\ \text{key } K_{AB}\!$

Attack against Denning-Sacco

 $\begin{array}{l} A \rightarrow S: \ A, B \\ S \rightarrow A: \ C_A, C_B \\ \hline A \rightarrow B: \ C_A, C_B, \{ \text{Sign}_A(\kappa_{AB}) \}_{K_B} \\ \hline \hline B \rightarrow S: \ B, C \\ S \rightarrow B: \ C_B, C_C \\ B \rightarrow C: \ C_A, C_C, \{ \text{Sign}_A(\kappa_{AB}) \}_{K_C} \\ \end{array}$ 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

More crypto protocols

Announcements intermission

More causes of crypto failure

Note to early readers

This is the section of the slides most likely to change in the final version

If class has already happened, make sure you have the latest slides for announcements

Outline

More crypto protocols

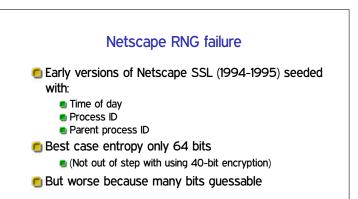
Announcements intermission

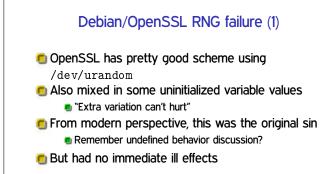
More causes of crypto failure

Random numbers and entropy

- Cryptographic RNGs use cipher-like techniques to provide indistinguishability
- Modern best practice: seed pool with 256 bits of entropy

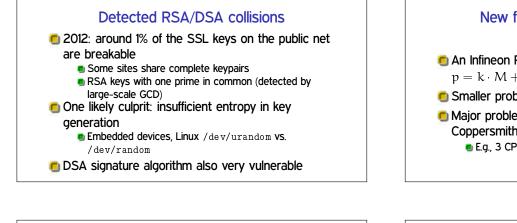
Suitable for security levels up to 2²⁵⁶





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



New factoring problem (CCS'17)

- **a** An Infineon RSA library used primes of the form $p = k \cdot M + (65537^{\alpha} \text{ mod } M)$
- Smaller problems: fingerprintable, less entropy
- Major problem: can factor with a variant of Coppersmith's algoritm
 - 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

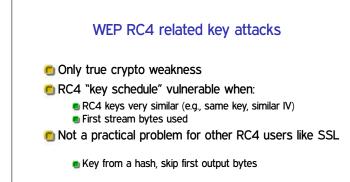
WEP "privacy"

- 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



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 π

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