

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 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

$A \rightarrow B : N_B, \{T_0, B, N_B\}_{K_B}$

- $A \rightarrow B$ : message sent from Alice intended for Bob
- B (after  $:$ ): Bob's name
- $\{\dots\}_K$ : 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
  - $A \rightarrow S : A, B$
  - key  $K_{AB}$ :  $S \rightarrow A : C_A, C_B$
  - $A \rightarrow B : C_A, C_B, \{\text{Sign}_A(K_{AB})\}_{K_B}$

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

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
- But rely on truly random seeding to stop brute force
  - Extreme case: no entropy → 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

## Newer factoring problem (CCS'17)

- An Infineon RSA library used primes of the form  $p = k \cdot M + (65537^a \bmod M)$
- 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

## 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

## Newer 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