Finite-State Analysis of SSL 3.0

John Mitchell Vitaly Shmatikov
Ulrich Stern

Stanford University

Murφ [Dill et al.]

- Describe finite-state system
  - State variables with initial values
  - Transition rules
  - Communication by shared variables
  - Scalable: choose system size parameters
- Specify correctness conditions
- Automatic exhaustive state enumeration

  Hash table to avoid repeating states

Success with research, industrial protocol verification

Applying Murφ to Security Protocols

[MMS97]

- Formulate protocol
- Add adversary
  - Full control over the “network”
  - Possible actions:
    - Intercept any message
    - Decrypt encrypted fields if key is known
    - Remember parts of messages
    - Generate new messages, using observed data and initial knowledge (e.g., public keys)
- State correctness conditions

Previous Results

- Short protocols (3-4 steps)
  - Needham-Schroeder
  - Kerberos
  - TMN
- Related work
  - FDR [Roscoe, Lowe et al.]
  - NRL Protocol Analyzer [Meadows]
  - Other tools [Kemmerer, Millen, …]

Protocol Analysis

- Protocol Specification

  Abstract notions of message, key, nonce, cryptographic functions

- Protocol Analysis

  High-level models for crypto primitives

- Protocol Implementation

  Specific key length, random number generator, encryption and decryption functions

Conventional Abstract Analysis

- Atomic interpretation of crypto primitives
  - Keys and nonces are atomic tokens
  - Decrypt only if key is known
  - No attacks involving cryptanalysis
    - No brute-force key search
    - No statistical tests on observed messages
    - No probabilities or partial information
- Messages *can* be split into components

What Do We Learn?

- Find an error
  - Error in Murφ model implies error in protocol
  - Can confirm error in implementation by testing
- Do not find an error
  - If implementation faithful to specification, no attack *within adversary model* will succeed
  - Increasing the number of participants may reveal an error

Failure to find error is not proof of correctness

Secure Socket Layer (*SSL*)

- De facto standard for Internet security

  Goal: “... provide privacy and reliability between two communicating applications”

- Handshake Protocol

  Use public-key cryptography to establish shared secret key

- Record Layer

  Transmit data using negotiated key

SSL Handshake Protocol

- Negotiate version, crypto suite
  - Possible “version rollback attack”
- Authenticate client and server
  - Appeal to “certificate authority”
- Use public key to establish shared secret

Several underlying primitives:

public key, signature, hash function, private key

Handshake Protocol Description

ClientHello $C \rightarrow S$ $C, Ver_C, Suite_C, N_C$

ServerHello $S \rightarrow C$ $Ver_S, Suite_S, N_S, sign_{CA}\{ S, K_S^+ \}$

ClientVerify $C \rightarrow S$ $sign_{CA}\{C, V_C\}$
$\{ Ver_C, Secret_C \}_{K_S^+}$
$sign_C \{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 +$
$Hash(Msgs + C + Master(N_C, N_S, Secret_C) + Pad_1)) \}$

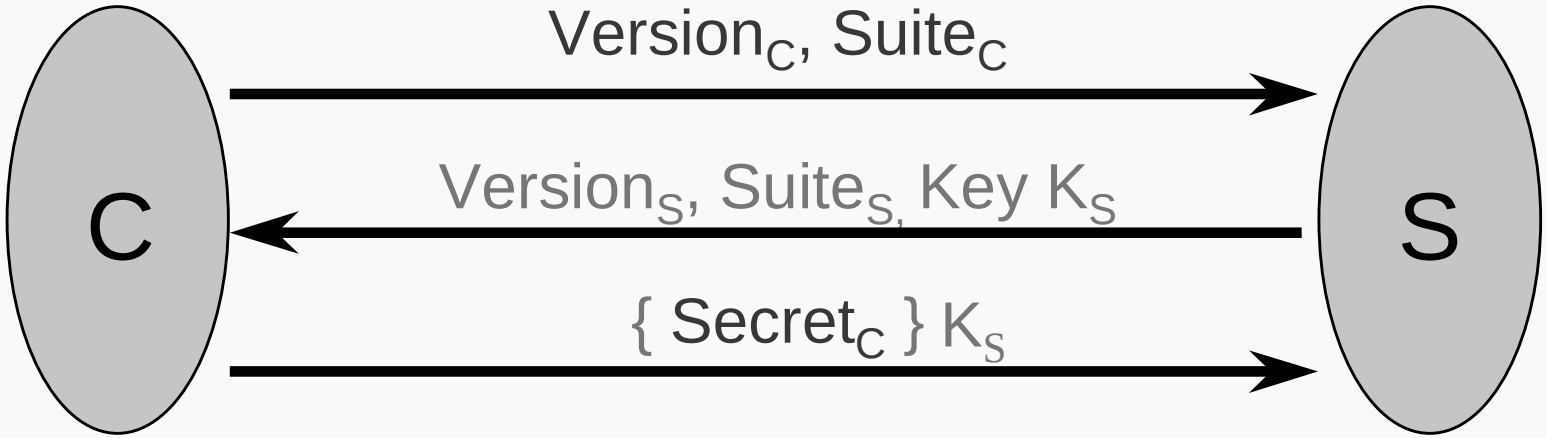
(Change to negotiated cipher)

ServerFinished $S \rightarrow C$ $\{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 +$
$Hash(Msgs + S + Master(N_C, N_S, Secret_C) + Pad_1))$
$\} Master(N_C, N_S, Secret_C)$

ClientFinished $C \rightarrow S$ $\{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 +$
$Hash(Msgs + C + Master(N_C, N_S, Secret_C) + Pad_1))$
$\} Master(N_C, N_S, Secret_C)$

Rational Reconstruction of *SSL*

- Begin with simple, intuitive protocol

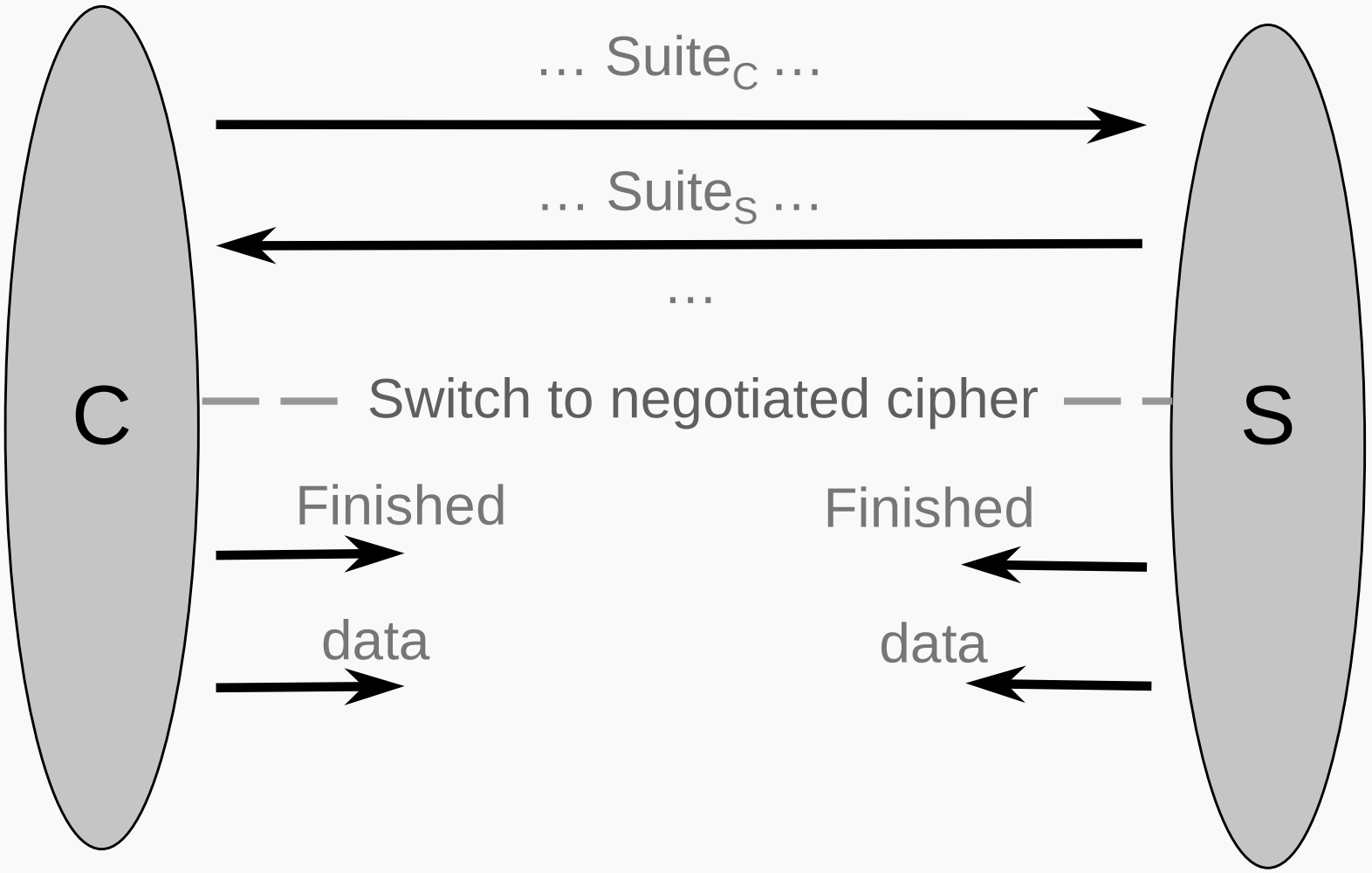


- Model check and find bug
- Add a piece of SSL to fix bug and repeat

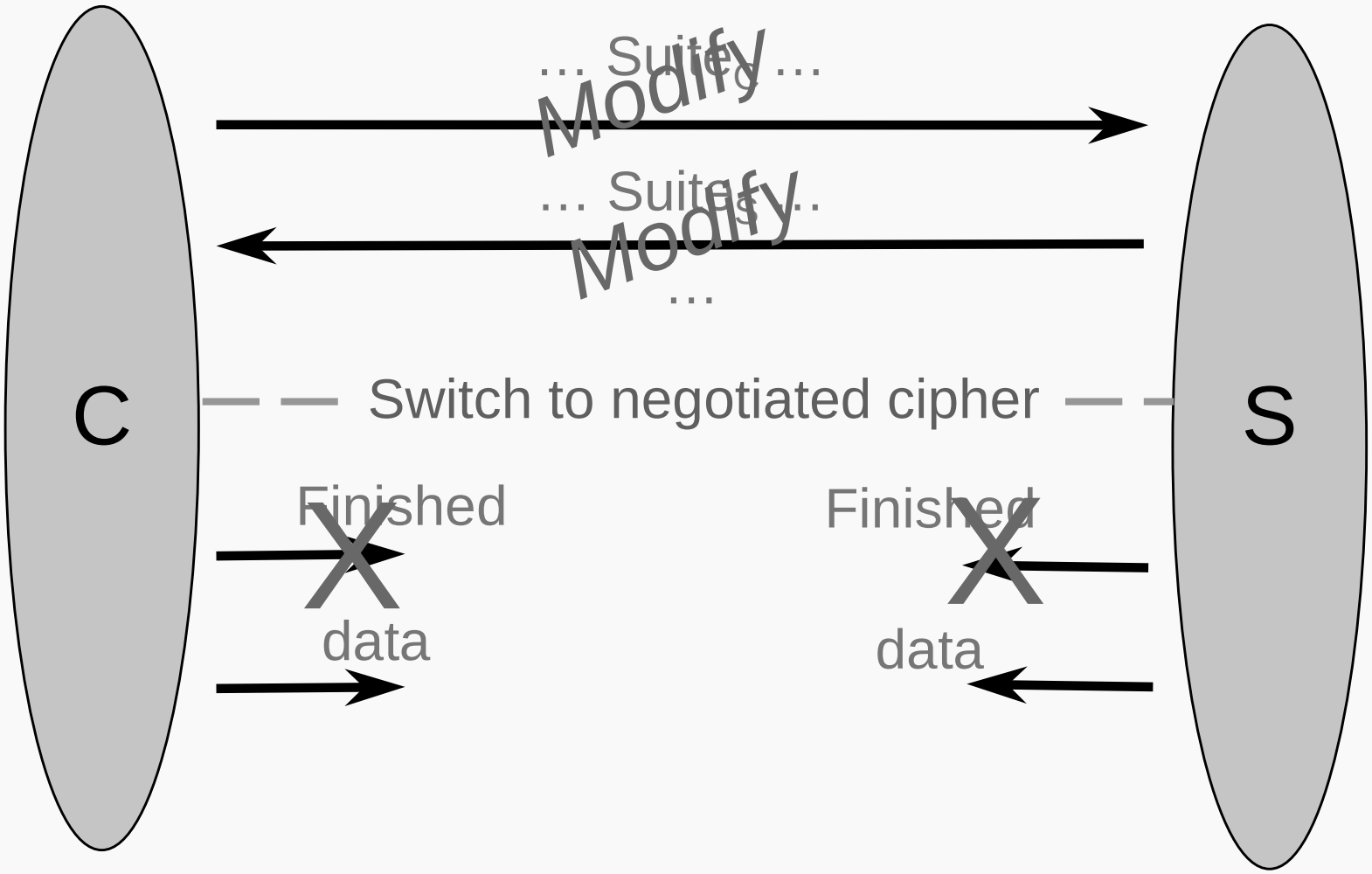
Summary of Reconstruction

- A = Basic protocol
- C = A + certificates for public keys
  - Authentication for client and server
- E = C + verification (Finished) messages
  - Prevention of version and crypto suite attacks
- F = E + nonces
  - Prevention of replay attacks
- Z = “Correct” subset of SSL

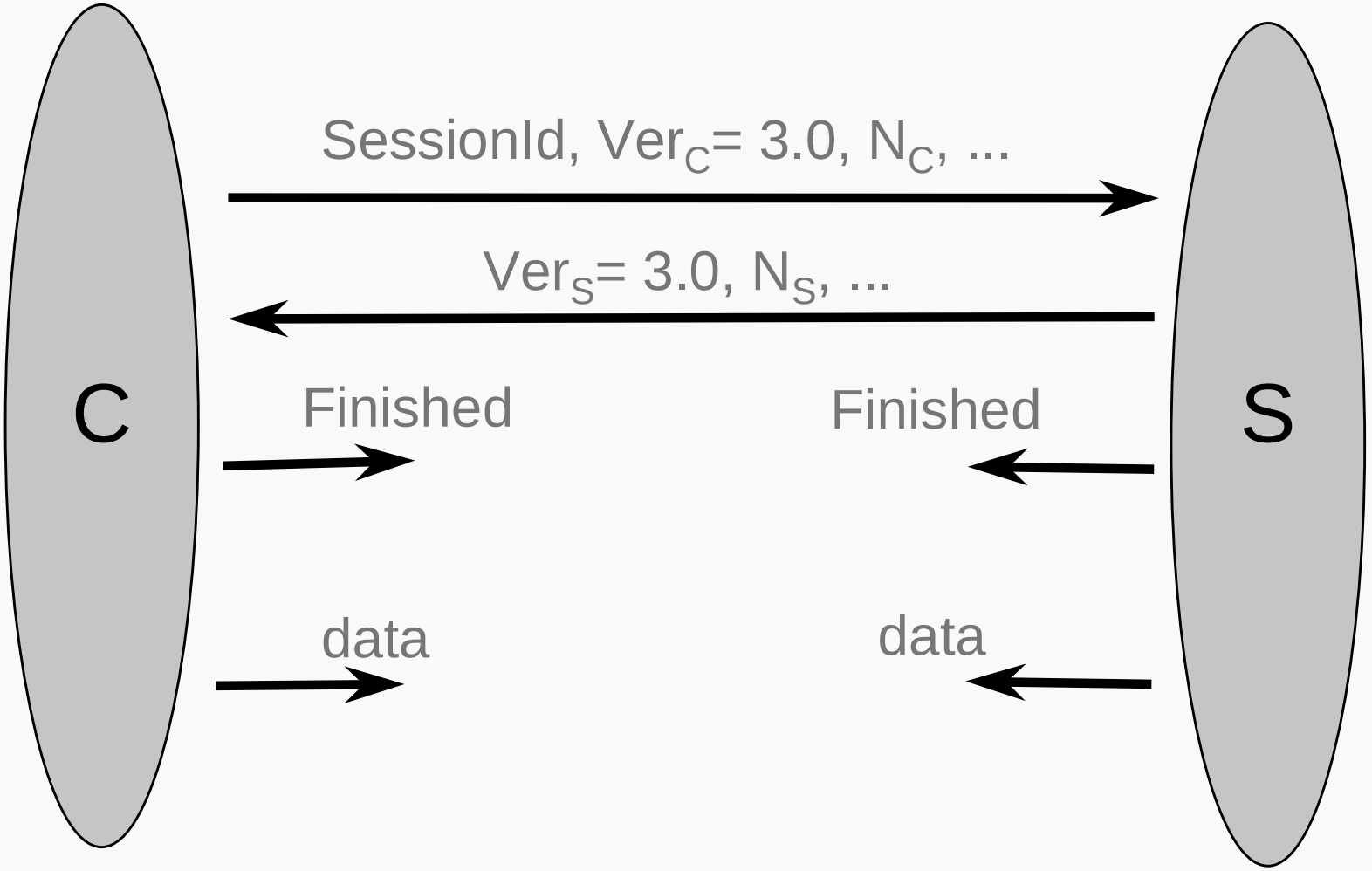
Anomaly (Protocol F)



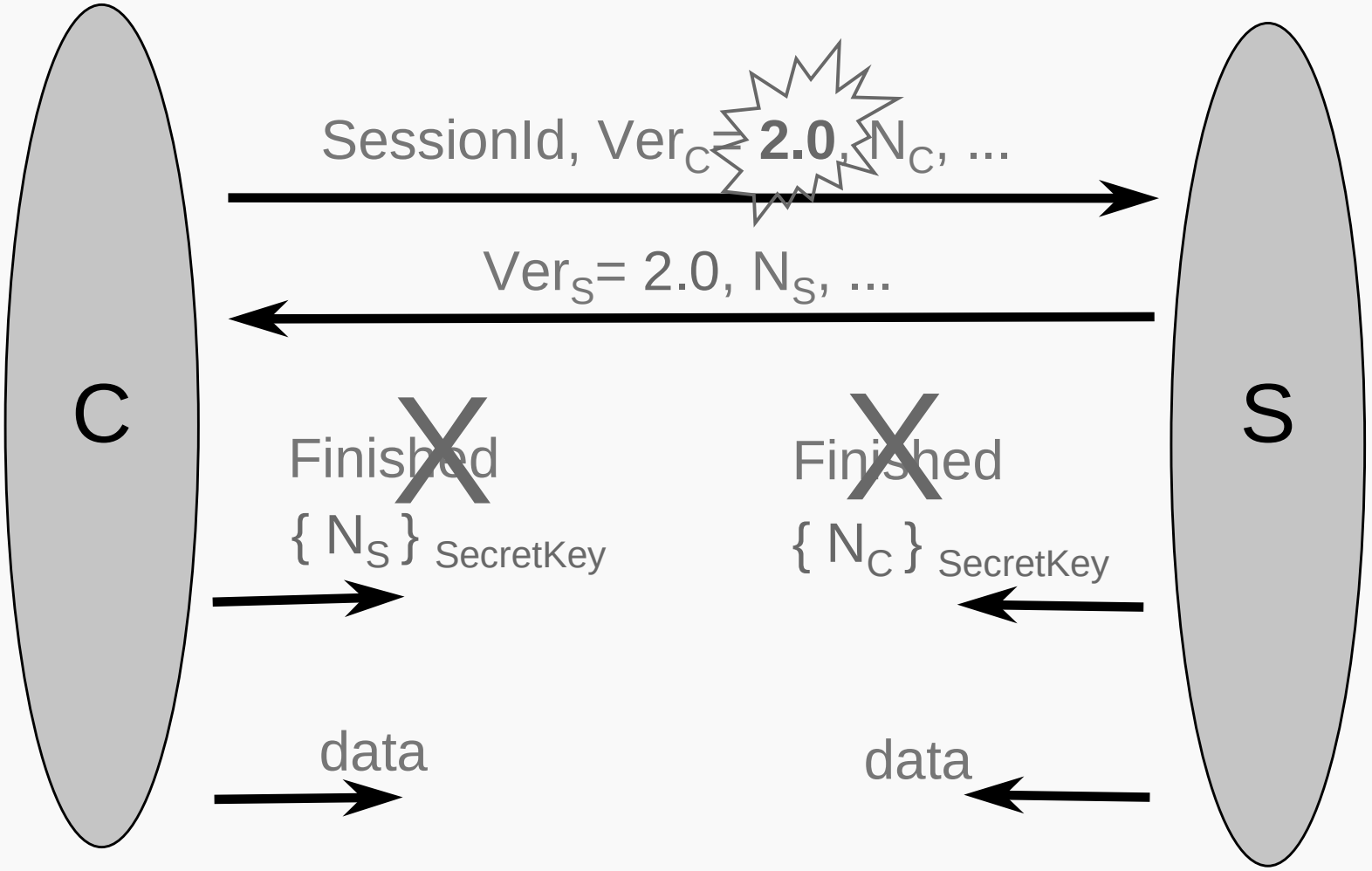
Anomaly (Protocol F)



Protocol Resumption



Version Rollback Attack



Conclusions

- Murφ is useful tool for complex protocols
- Rational reconstruction of protocol
  - Understand protocol
  - Ensure “completeness” of analysis
  - Protocol spec simpler, more precise than RFC
- Uncover problem areas in SSL
  - SSL 2.0 errors identified
  - “Gray” areas in the resumption protocol

Limitations

- System size with current methods
  - 2-6 participants

    SSL: 2 clients, 1 server, 2 simultaneous sessions
  - 6-8 steps in protocol
  - State reduction techniques
- Adversary model
  - Cannot model cryptanalysis
  - Cannot model computational effort