# Analysis of EAP-GPSK Authentication Protocol

Protocol eXchange
October 2, 2008

John C. Mitchell, Arnab Roy[1]
Paul Rowe, Andre Scedrov[2]
[1]Stanford University
[2]University of Pennsylvania

# Purpose of Talk

Results of analysis of the IETF EAP-GPSK Protocol

- Found 3 weaknesses in version 5
  - Repairable client-side DoS Attack
  - Anomaly with key derivation function
  - Ciphersuite downgrading attack
- Resulted in discussions with IETF EMU working group

# Outcome of IETF EMU Discussions

Main focus of this talk

- Provided a solution for the DoS attack
  - Resulted in a change in the message structure
  - Our biggest contribution to the protocol
- Discussion of key derivation anomaly
  - We suggested there might be a problem
  - No concrete attack
  - Working group developed a fix
- Discussion of downgrading attack
  - Resulted in additional comments and requirements in the specification

# Previous Work

- Many analyses exist for well-established protocols:
  - Kerberos [MMS97],[BP98],[CJSTW06]
  - SSL [MSS98]
  - 802.11i [HM04]
  - etc.
- In some cases analysis occurs after standardization
- Our analysis was integrated with the standardization process
  - Makes it easier to make necessary changes

# Extensible Authentication Protocol

- Framework designed to support a variety of authentication methods
- Designed for data link layer
  - Does not assume IP connectivity
- Supports
  - Duplicate Elimination
  - Retransmission
- Assumes lower layers properly order packets

# Extensible Authentication Protocol

- Works on
  - PPP connections
  - IEEE 802 wired and wireless LAN networks
  - Over the Internet
- Three phases
  - Discovery
  - Authentication
  - Secure Association

# Generalized PreShared Key

- Protocol Goals:
  - Mutual authentication
  - Key agreement
- Designed to be lightweight and efficient
  - Uses symmetric cryptography minimizing computational requirements
  - Minimizes the number of EAP rounds
- Designed to be flexible
  - Allows for negotiation of ciphersuites

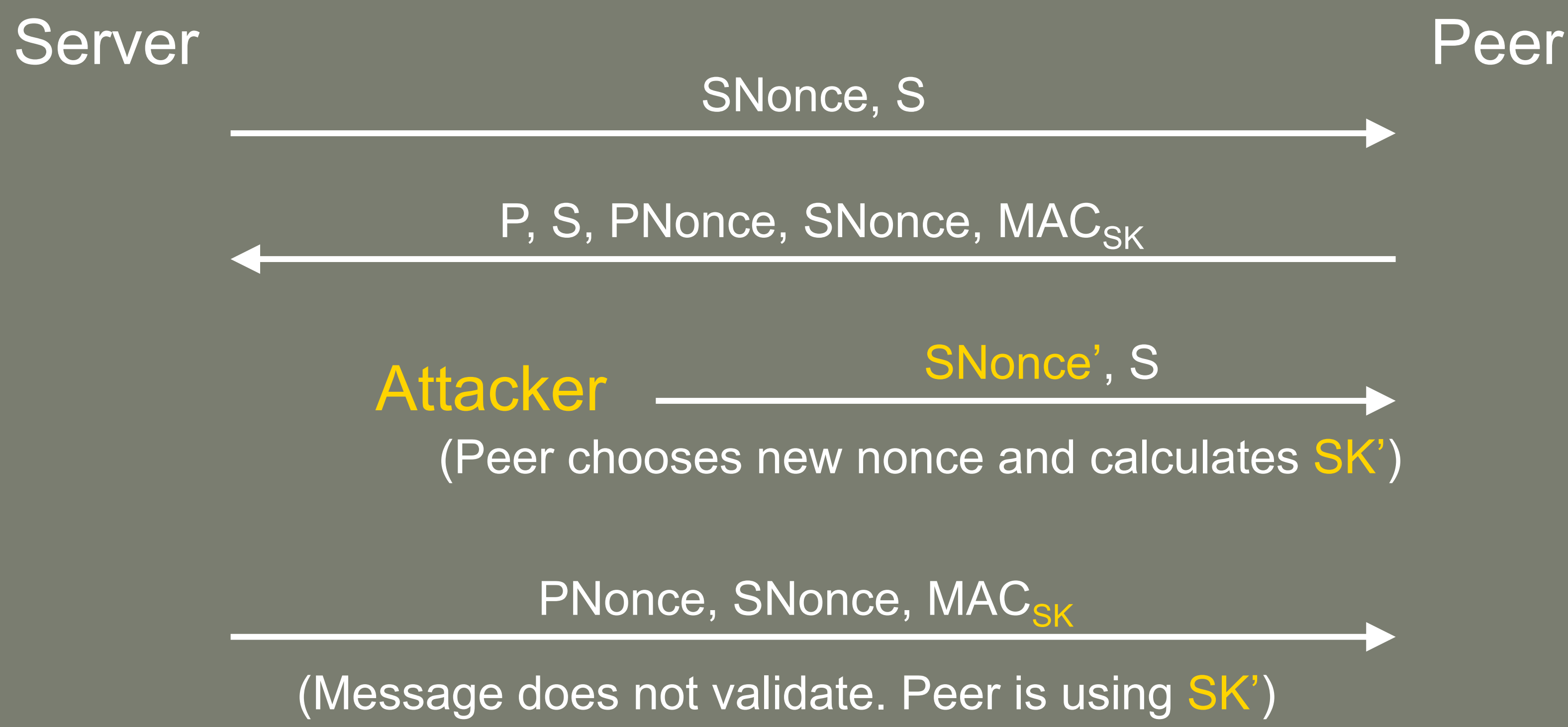
# EAP-GPSK


- inputString = PNonce || P || SNonce || S
- MK = KDF-KS(0x00, PL || PSK || CSuiteSel || inputString)[0..KS-1]
- SK = KDF-{128+2KS}(MK, inputString)[128..127+KS]
- PK = KDF-{128+2KS}(MK, inputString)[128+KS..127+2*KS]

# Client-Side DoS Attack

- Virtually identical to an attack found in 802.11i 4-Way Handshake [HM04]
- Memory exhaustion attack
  - Particularly worrisome for small devices
- Results in irreconcilable discrepancy between Peer's and Server's session keys
- Occurs due to Peer allocating memory based on unauthenticated *Message 1*

# Client-Side DoS Attack



# What Goes Wrong?

- *Message 1* is unauthenticated
  - Typical for purely symmetric key protocols
  - Avoids using long term secret as a key
- Peer allocates memory in response to unauthenticated *Message 1*
  - Allows attacker to fill Peer's memory
- Valid session information is irrecoverable
- A solution should either:
  - Introduce message authentication earlier
  - Restrict Peer's actions in response to unauthenticated *Message 1*

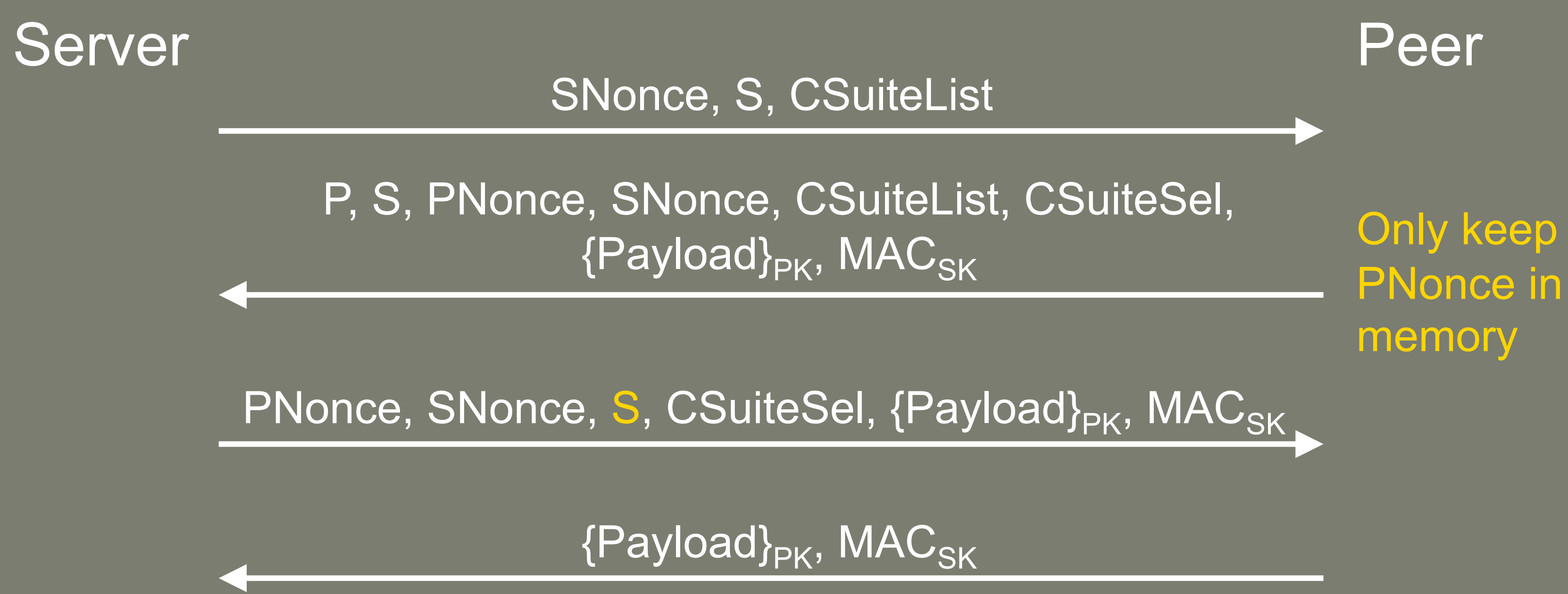
# Why Accept Duplicate *Message 1*?

- No delivery guarantee
  - A repeated *Message 1* may indicate that *Message 2* never arrived
  - If Peer does not accept, the protocol reaches deadlock without attacker
- Otherwise easier attack is possible
  - Attacker sends fake *Message 1* prior to any communication to block the protocol
  - Does not require precise timing

# DoS Fix

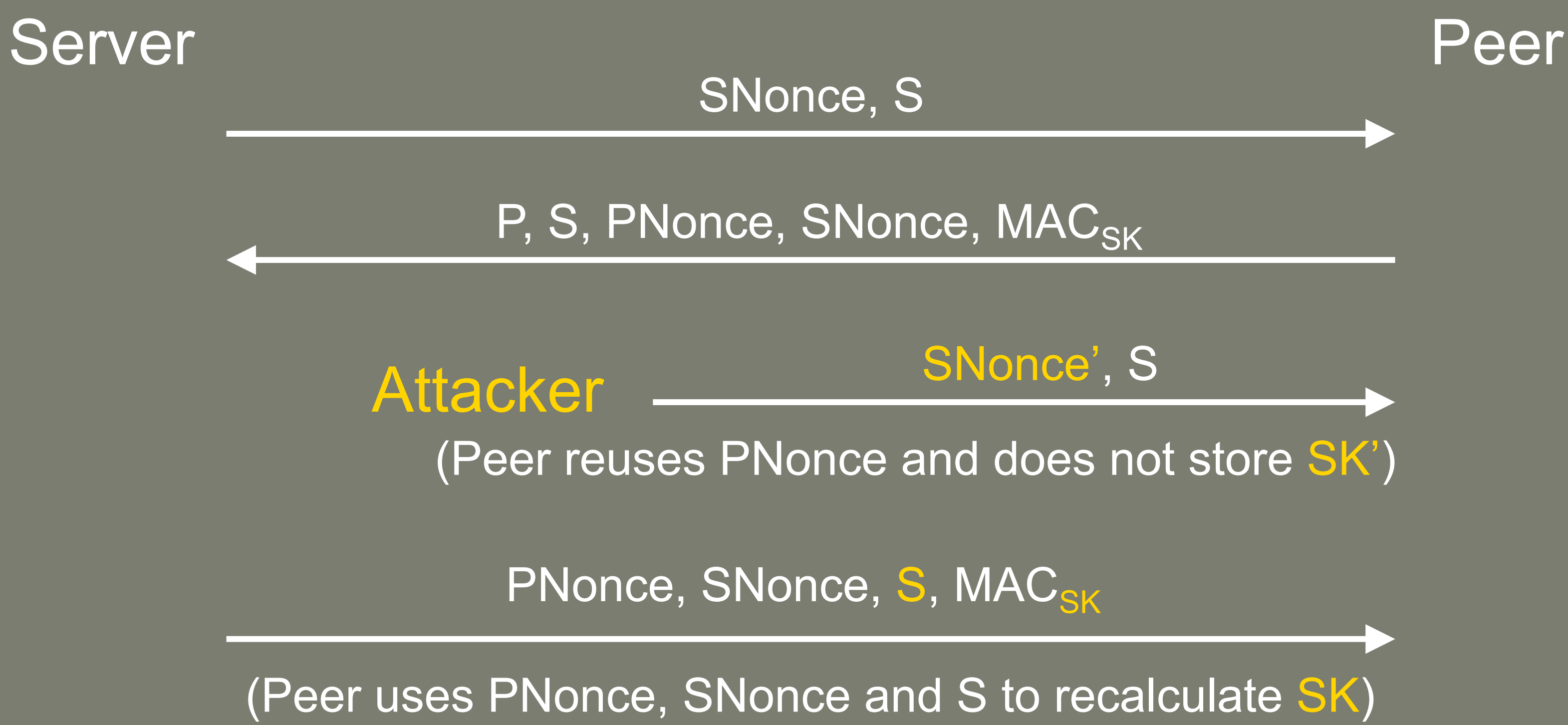
- Peer stores only PNonce when receiving *Message 1*
- When receiving duplicate *Message 1*, Peer reuses PNonce
  - Peer now maintains state per Server instead of state per session
- Modify *Message 3* so Peer can recalculate the key SK from its contents
- This is similar to the fix ultimately adopted by the IEEE 802.11i working group [HM04]

# DoS Fix




- inputString = PNonce || P || SNonce || S
- MK = KDF-KS(0x00, PL || PSK || CSuiteSel || inputString)[0..KS-1]
- SK = KDF-{128+2KS}(MK, inputString)[128..127+KS]
- PK = KDF-{128+2KS}(MK, inputString)[128+KS..127+2*KS]

# Protection Against DoS Attack




(Only relevant message fields are shown)

# Key Derivation

- inputString = PNonce || P || SNonce || S
- MK = KDF-KS(0x00, PL || PSK || CSuiteSel || inputString)[0..KS-1]
- SK = KDF-{128+2KS}(MK, inputString)[128..127+KS]
- PK = KDF-{128+2KS}(MK, inputString)[128+KS..127+2*KS]

- Not our area of expertise
- Use of 0x00 as key might reduce entropy of MK
- This usage is problematic for modeling reasons

# Modeling Considerations

- If SK and PK are to have high entropy we can assume
  - KDF is a PRF
  - MK has high entropy
- Let $kdf_{00}(y)$ = KDF(0x00, y)
- 3 natural modeling assumptions:
  - $kdf_{00}$ is a random oracle
    - Very strong assumption
  - KDF is a PRF
    - $kdf_{00}(y)$ could still have low entropy
  - $kdf_{00}$ is a PRG
    - Input is structured and partially known

# Working Group's Solution

- Use PSK as the key to KDF
- This is familiar from TLS
  - We model KDF as a PRF
- Designers want to support variety of key lengths
  - Require minimum length keys
  - Truncate long keys when using KDF that takes shorter keys

# Downgrading Attack

- Attacker modifies CSuiteList in *Message 1*
  - If Peer chooses a ciphersuite with a weak MAC, authentication and secrecy could fail
- Specification contains one mandatory ciphersuite
  - Guarantees a strong ciphersuite is always available
- Ciphersuites not required to support encryption
  - Unwitting Peer may choose a ciphersuite with no encryption
  - $\{Payload\}_{PK}$ is vulnerable before CSuiteList is Authenticated

# Result of Discussions

- EMU working group added comments to specification
  - New ciphersuites required to meet integrity standards
  - Method extensions relying on confidentiality must make this clear
  - Peer must not transmit confidential information in *Message 2* $\{\text{Payload}\}_{PK}$

# Murφ in the Analysis

- We used the model checker Murφ to aid our analysis
- Murφ can help find flaws but cannot demonstrate protocol correctness
- Previous experience with Murφ helped in discovering weakness by hand
- Both DoS and downgrading attacks were detected by our Murφ model

# Protocol Composition Logic

- PCL is a logic of authentication [DDMR07]
  - Proven sound for runs with any number of principals and sessions
- Model protocols by specifying roles in a language based on *cords*
- Uses formulas of the form $\theta[P]_X\varphi$
  - If $\theta$ holds before actions P then $\varphi$ will hold afterwards
- Helps to clarify assumptions about crypto primitives

# What We Prove (Informally)

- Theorem (Secrecy): After a run of the protocol the keys PK and SK are known only to Server and Peer.

- Theorem (Authentication): After a run of the protocol the Server and Peer have matching records of the run.

# Conclusion

- Protocols are still being designed with familiar flaws
- Integrating protocol analysis with development lets us catch these flaws *before* widespread deployment
- Interaction with standardization bodies brings awareness of common flaws to protocol designers

Thank You