Proof of Chain of Possession (POCOP) Tokens

Abstract

This document introduces a POCOP mechanism based on nested, chained HMACs constructions to provide chained authenticity and integrity protection.

Introduction

Bearer tokens are vulnerable at rest and in transit when an attacker is able to intercept a token to illegally access private information. In order to mitigate some of the risk associated with bearer tokens, proof-of-chain-of-possession may be used to authenticate the token. Chain-of-possession token is a chronological tamper-resistant record of all its possessors and the changes that have been made to it.

Concept

Chained-MACs-with-Multiple-Messages

The MAC_{final} value is calculated starting with HMAC root key K_{root} and the first message m_1, each MAC value being used as the HMAC key for the next message.

MAC_{final} = HMAC(...HMAC(HMAC(K_{root}, m_1), m_2,), ...m_n)

Google Macaroons are based on this construction.

Chained-MACs-with-Multiple-Keys

The MAC_{final} value is calculated starting with the first HMAC key K_1 and the root message m_{root}, each MAC value being used as the HMAC message for the next Key.

MAC_{final} = HMAC(K_n, ...HMAC(K_2, HMAC(K_1, m_{root})))

This construction provides the basis of the POCOP mechanism.

Example 1 of Complex Chained Construction

MAC_{final} = HMAC(K_n, ...HMAC(HMAC(K_2, HMAC(HMAC(K_1, m_1), m_2)), ...m_n))

Example of client chaining with RS_1 and RS_2:

MAC_{final} = HMAC(K_{RS_2}, HMAC(HMAC(K_{RS_1}, HMAC(HMAC(K_{client}, m_{client}), m_{RS_1})), m_{RS_2}))

broken down into individual MACs

MAC = HMAC(K_{client}, m_{client})

MAC = HMAC(MAC, m_{RS_1})

MAC = HMAC(K_{RS_1}, MAC)

MAC = HMAC(MAC, m_{RS_2})

MAC_{final} = HMAC(K_{RS_2}, MAC)

This complex construction with multiple messages and multiple keys is applicable to the JWT format.

Example 2 of Complex Chained Construction

MAC_{AS} = HMAC(K_{AS}, $NONCE_{AS}$ || m_{AS})

*

MAC_{AS} = HMAC(K_{client}, MAC_{AS})

MAC_{client} = HMAC(K_{client}, MAC_{AS} || $NONCE_{client}$ || m_{client})

*

MAC_{client} = HMAC(K_{RS_1}, MAC_{client})

MAC_{RS_1} = HMAC(K_{RS_1}, MAC_{client} || $NONCE_{RS\_1}$ || m_{RS_1})

*

$MAC_{RS\_1} = HMAC(K_{RS\_2}, MAC_{RS\_1})$

$MAC_{final} = HMAC(K_{RS\_2}, MAC_{RS\_1} \| NONCE_{RS\_2} \| m_{RS\_2})$

Intermediate Conclusion

Nested, chained complex HMACs constructions applied on tokens, tickets, cookies and macaroons may be used to implement both new authorization protocols and to enhance existing ones.

POCOP Token Mechanism

The Chained-MACs-with-Multiple-Keys construction is used as the basis of the POCOP token mechanism.

The root message of the token must contain:

- The random NONCE to prevent replay attack.
- The claim that identifies who created the token.
- The timestamp of when the token was issued.

The claims can be chained using the Chained-MACs-with-Multiple-Messages construction. The complex combination of Chained-MACs-with-Multiple-Messages and Chained-MACs-with-Multiple-Keys constructions forms a basis of the Auditable Authorization mechanism.

Conclusion

...

Acknowledgment

Credits go to WG - User-Managed Access and Google Research Publications.