Formal Verification of EDHOC

IETF Interim Meeting

Theis Grønbech Petersen, Thorvald Jørgensen, Alessandro Bruni, Carsten Schürmann

5-3-2019

Trust, but Verify

- What security properties?
- Comparison with other protocols, e.g. TLS 1.3?
- Continuous verification

Security Properties

- Identity protection (for client and server)
  - Running the protocol does not reveal the identity of the participants
- Secrecy of session keys and data
  - Session keys and application data are known only by the client and server running the protocol
- Perfect forward secrecy
  - If long-term keys are leaked after running the protocol, the session keys are still not compromised
- Session independence
  - Compromising specific session keys will not affect other sessions
- (Weak) Post compromise security
  - An attacker with access to an oracle that allows encryption and signing using long term keys cannot interfere the protocol once access to the oracle is removed

The Usual Caveats

- Symbolic model: abstract modeling of crypto, automated verification
- Dolev-Yao attacker
  - Attacker has control over the communication channel, can drop, inject, replay and construct their own messages
  - Cryptography as a blackbox: considered perfect and unbreakable
- Key leakage
  - We relax the Dolev-Yao model by revealing all long-term keys (PFS) and session keys (Session Independence)
- No ciphersuite negotiation

EDHOC Asymmetric (draft 08)

Initiator (U)
Knows g, U, APP_1, APP_3

Responder (V)
Knows g, V, APP_2

Initiator (U):

Generates $S_U,\ N_U,\ x$
$E_U = g^x$

$U \rightarrow V$: $msg_1 : 1, S_U,\ N_U,\ E_U,\ ALG_1,\ APP_1$

Responder (V):

Generates $S_V,\ N_V,\ y$
$E_V = g^y$
$aad_2 = H(msg_1, data_2)$
$K_2 = H_{KDF}({E_U}^y,\ aad_2)$

$V \rightarrow U$: $msg_2 : \overbrace{2, S_U, S_V, N_V, E_V, ALG_2}^{data_2}, aead_{K_2}^{aad_2}(sign_V(ID_V, aad_2, APP_2))$

Initiator (U):

$K_2 = H_{KDF}({E_V}^x,\ aad_2)$
$aad_3 = H(H(msg_1, msg_2), data_3)$
$K_3 = H_{KDF}({E_V}^x,\ aad_3)$

$U \rightarrow V$: $msg_3 : \overbrace{3, S_V}^{data_3}, aead_{K_3}^{aad_3}(sign_U(ID_U, aad_3, APP_3))$

Responder (V):

$K_3 = H_{KDF}({E_U}^y,\ aad_3)$

EDHOC Symmetric (draft 08)

Initiator (U)
Knows g, PSK, APP_1, APP_3

Responder (V)
Knows g, PSK, APP_2

Initiator: Generates $S_U,\ N_U,\ x$
$E_U = g^x$

$msg_1 : 4,\ S_U,\ N_U,\ E_U,\ ALG_1,\ KID,\ APP_1$ (Initiator → Responder)

Responder: Generates $S_V,\ N_V,\ y$
$E_V = g^y$
$aad_2 = H(msg_1, data_2)$
$K_2 = H_{KDF}({E_U}^y,\ aad_2,\ PSK)$

$msg_2 : \overbrace{5,\ S_U,\ S_V,\ N_V,\ E_V,\ ALG_2}^{data_2},\ enc_{K_2}^{aad_2}(APP_2)$ (Responder → Initiator)

Initiator:
$K_2 = H_{KDF}({E_V}^x,\ aad_2,\ PSK)$
$aad_3 = H(H(msg_1, msg_2), data_3)$
$K_3 = H_{KDF}({E_V}^x,\ aad_3,\ PSK)$

$msg_3 : \overbrace{6,\ S_V}^{data_3}, aead_{K_3}^{aad_3}(APP_3)$ (Initiator → Responder)

Responder: $K_3 = H_{KDF}({E_U}^y,\ aad_3,\ PSK)$

Discoveries (TL;DR)

Discoveries (TL;DR)

- It's a SIGMA-I protocol

Discoveries (TL;DR)

- It's a SIGMA-I protocol
- Weak guarantees for APP_2 in Asymmetric Mode

Discoveries (TL;DR)

- It's a SIGMA-I protocol
- Weak guarantees for APP_2 in Asymmetric Mode
- Authentication, secrecy (of keys, application data)
- Perfect forward secrecy (weaker guarantees for active attacks)

EDHOC Evolution (draft-08 $\rightarrow$ draft-11)

- Discussion on APP_2: removal, reintroduction, renaming
- Removal of nonces

Draft-11 verification (WIP)

- Stronger attacker model:
  - malicious principals with registered keys
  - session independence (revealing session keys should maintain all the checked properties for other sessions)

Draft-11 verification (WIP)

- Stronger attacker model:
  - malicious principals with registered keys
  - session independence (revealing session keys should maintain all the checked properties for other sessions)
- Results:

Draft-11 verification (WIP)

- Stronger attacker model:
  - malicious principals with registered keys
  - session independence (revealing session keys should maintain all the checked properties for other sessions)
- Results:
  - nothing surprising (fortunately)
  - w/ session independence authentication proofs running for 10+ days

Verification results (Identity protection)

```
RESULT attacker(idI(pk(skU[!1 = v_4063])))
  ==> event(LTK_Reveal(skU[!1 = v_4061]))
    ||   event(SessK3A_Reveal(x_180,skU[!1 = v_4062])) is true.
RESULT attacker(idR(pk(skU[!1 = v_28705])))
  ==> event(LTK_Reveal(skU[!1 = v_28703]))
    ||  event(SessK2A_Reveal(x_184,skU[!1 = v_28704])) cannot be proved.
RESULT attacker_p1(idI(pk(skU[!1 = v_54380])))
  ==> event(LTK_Reveal(skU[!1 = v_54378]))
    ||  event(SessK3A_Reveal(x_188,skU[!1 = v_54379])) is true.
RESULT attacker_p1(idR(pk(skU[!1 = v_78966])))
  ==> event(LTK_Reveal(skU[!1 = v_78964]))
    ||  event(SessK2A_Reveal(x_192,skU[!1 = v_78965])) cannot be proved.
```

Verification results (Secrecy)

```
RESULT attacker(APP_2A(pk(skU_205),skV_206,S_V_207,K_2_208))
  ==> event(LTK_Reveal(skU_205))
    ||  event(SessK2A_Reveal(K_2_208,skU[!1 = v_178396])) cannot be proved.
RESULT attacker(APP_2A'(pk(skU_210),skV_211,S_V_212,K_2_213))
  ==> event(LTK_Reveal(skU_210))
    ||  event(SessK2A_Reveal(K_2_213,skU[!1 = v_204184])) is true.
RESULT attacker(APP_3A(skU_215,pk(skV_216),S_U_217,K_3_218))
  ==> event(LTK_Reveal(skV_216))
    ||  event(SessK3A_Reveal(K_3_218,skU[!1 = v_228771])) is true.
RESULT attacker(APP_2S(PSK_196,S_U_197,K_2_198))
  ==> event(PSK_Reveal(PSK_196))
    ||  event(SessK2S_Reveal(K_2_198)) is true.
RESULT attacker(APP_2S'(PSK_199,S_U_200,K_2_201))
  ==> event(PSK_Reveal(PSK_199))
    ||  event(SessK2S_Reveal(K_2_201)) is true.
RESULT attacker(APP_3S(PSK_202,S_V_203,K_3_204))
  ==> event(PSK_Reveal(PSK_202))
    ||  event(SessK3S_Reveal(K_3_204)) is true.
```

Verification results (Perfect Forward Secrecy)

```
RESULT attacker_p1(APP_2A(pk(skU_220),skV_221,S_V_222,K_2_223))
  ==> event(LTK_Reveal(skU_220))
    ||  event(SessK2A_Reveal(K_2_223,skU[!1 = v_253358])) cannot be proved.
RESULT attacker_p1(APP_2A'(pk(skU_225),skV_226,S_V_227,K_2_228))
  ==> event(LTK_Reveal(skU_225))
    ||  event(SessK2A_Reveal(K_2_228,skU[!1 = v_279151])) is true.
RESULT attacker_p1(APP_3A(skU_230,pk(skV_231),S_U_232,K_3_233))
  ==> event(LTK_Reveal(skV_231))
    ||  event(SessK3A_Reveal(K_3_233,skU[!1 = v_303742])) is true.
RESULT attacker_p1(APP_2S(PSK_235,S_U_236,K_2_237))
  ==> event(PSK_Reveal(PSK_235))
    ||  event(SessK2S_Reveal(K_2_237)) cannot be proved.
RESULT attacker_p1(APP_2S'(PSK_238,S_U_239,K_2_240))
  ==> event(PSK_Reveal(PSK_238))
    ||  event(SessK2S_Reveal(K_2_240)) is true.
RESULT attacker_p1(APP_3S(PSK_241,S_V_242,K_3_243))
  ==> event(PSK_Reveal(PSK_241))
    ||  event(SessK3S_Reveal(K_3_243)) is true.
```

Verification results (Authentication)

```
RESULT inj-event(midInitiatorA(U_253,V_254,E_V_255))
  ==> inj-event(startResponderA(U',V_254,E_V_255,skV_256))
    ||  event(LTK_Reveal(skV_256)) is true.
RESULT inj-event(endResponderA(U_257,V_258,E_U_259))
  ==> inj-event(startInitiatorA(U_257,V_258,E_U_259,skU_260))
    ||  event(LTK_Reveal(skU_260)) is true.
RESULT inj-event(endInitiatorA(U_261,V_263,E_V_264))
  ==> inj-event(startResponderA(U'_262,V_263,E_V_264,skV_265))
    ||  event(LTK_Reveal(skV_265)) is true.
```

Result table (draft 08)

Variant	Data	Secrecy	(at completion)	PFS	(at completion)	Integrity	(at completion)
Asymmetric	APP_1	–	–	–	–	✗	✓
	APP_2	✗	✓	✗	✓	✓	✓
	APP_3	✓	✓	✓	✓	✓	✓
Symmetric	APP_1	–	–	–	–	✗	✓
	APP_2	✓	✓	✗	✓	✓	✓
	APP_3	✓	✓	✓	✓	✓	✓

Comparison with state of the art verification

Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate

Karthikeyan Bhargavan, Bruno Blanchet, Nadim Kobeissi
INRIA
{karthik.bhargavan,bruno.blanchet,nadim.kobeissi}@inria.fr

Secrecy: If an application data message m is sent over a session cid between an honest client C and honest server S, then this message is kept confidential from an attacker who cannot break the cryptographic constructions used in the session cid.

Forward Secrecy: Secrecy (above) holds even if the long-term keys of the client and server (sk_C, pk_C, psk) are given to the adversary after the session cid has been completed and the session keys k_c, k_s are deleted by C and S.

Authentication: If an application data message m is received over a session cid from an honest and authenticated peer, then the peer must have sent the same application data m in a matching session (with the same parameters $cid, offer_C, mode_S, pk_C, pk_S, psk, k_c, k_s, psk'$).

Replay Prevention: Any application data m sent over a session cid may be accepted at most once by the peer.

Unique Channel Identifier: If a client session and a server session have the same identifier cid, then all other parameters in these sessions must match (same $cid, offer_C, mode_S, pk_C, pk_S, psk, k_c, k_s, psk'$).

Status

- Some proofs are WIP for draft-11
- No surprises

Status

- Some proofs are WIP for draft-11
- No surprises
- Questions?