A Comprehensive Formal Security Analysis of OPC UA

Vincent DIEMUNSCH, Lucca HIRSCHI & Steve KREMER

French Cybersecurity Agency (ANSSI) & Université de Lorraine, CNRS, Inria, LORIA, France

15th September 2025

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

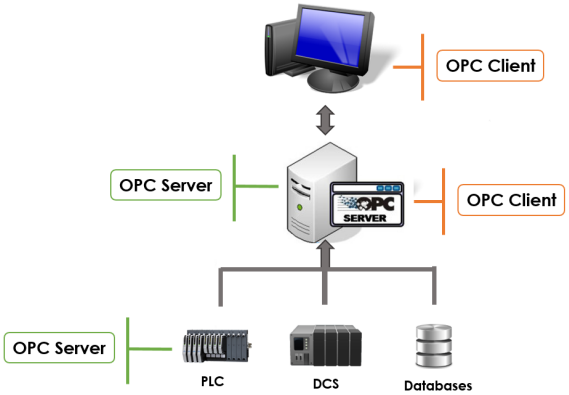
Inria

The protocol OPC Unified Architecture (IEC 62541, v1.04)

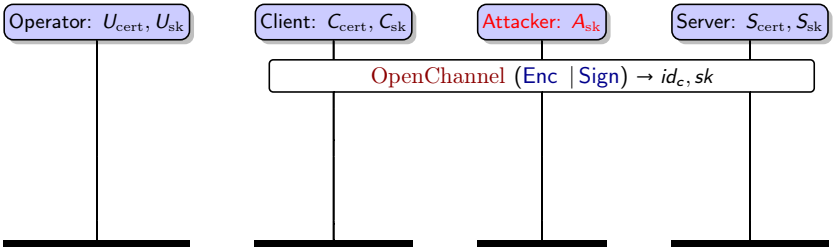
An Industrial Control System (ICS) protocol

- to command and control automation devices
- standardized as IEC 62541
- with built-in cybersecurity

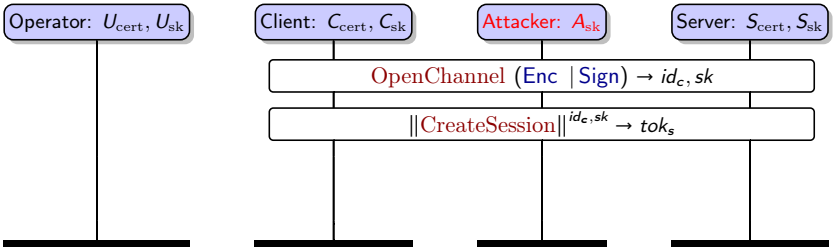
Recommended for critical infrastructures
(e.g., power grids, air traffic control, oil & gaz utilities, . . .)



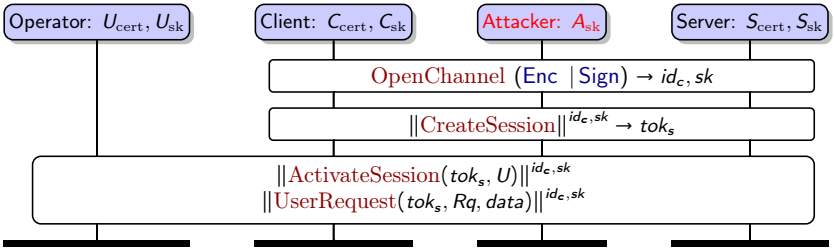
An overview of OPC UA Binary



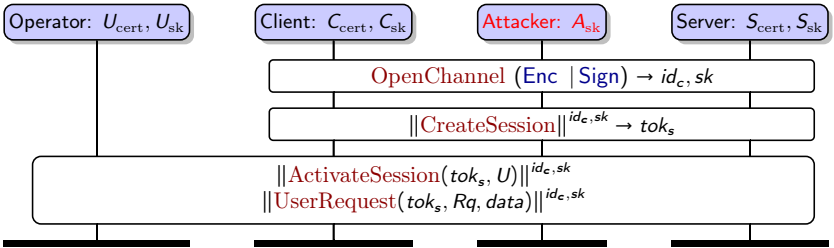
An overview of OPC UA Binary

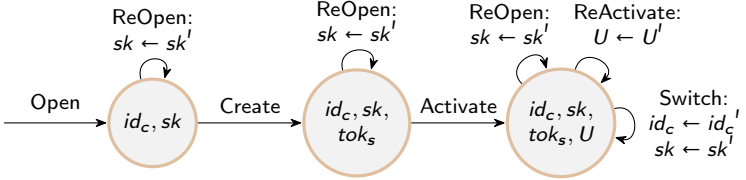


An overview of OPC UA Binary



An overview of OPC UA Binary





Is OPC UA secure?

What about authentication and confidentiality of messages

- when an active attacker is on the ICS network,
- with potentially compromised long-term keys?

Is OPC UA secure?

What about authentication and confidentiality of messages

- when an active attacker is on the ICS network,
- with potentially compromised long-term keys?

Related work:

- *OPC UA Security Analysis* of German BSI in 2022
- 2 formal analyzes of v1.03 on partial sub-protocols:
  1. ProVerif: channel establishment, RSA only [Puys et al., SAFECOMP 2016]
  2. Tamarin: flow integrity of secure channel [Dreier et al., Computers & Security 2019]

Is OPC UA secure?

What about authentication and confidentiality of messages

- when an active attacker is on the ICS network,
- with potentially compromised long-term keys?

Related work:

- *OPC UA Security Analysis* of German BSI in 2022
- 2 formal analyzes of v1.03 on partial sub-protocols:
  1. ProVerif: channel establishment, RSA only [Puys et al., SAFECOMP 2016]
  2. Tamarin: flow integrity of secure channel [Dreier et al., Computers & Security 2019]

Contributions:

- a comprehensive formal model of v1.05 up to user sessions in both RSA and ECC
- automated proofs of security properties
- 5 new attacks and 3 weaknesses responsibly disclosed to the OPC Foundation

ProVerif: a cryptographic protocol verifier

proverif.inria.fr

Symbolic model:

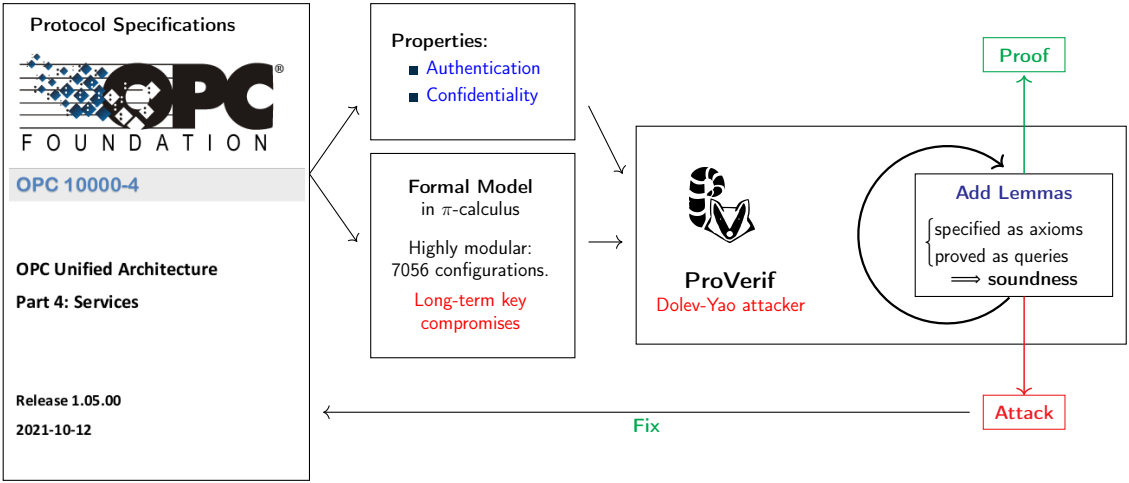
- protocol specified as parallel processes (applied π-calculus)
- messages are terms: $sign(C_{\text{cert}}, enc(nonce, S_{\text{pk}}), C_{\text{sk}})$
- Dolev-Yao attacker:
  - can intercept messages
  - can forge and inject messages from known terms
  - cryptography is idealized: $dec(enc(m, C_{\text{pk}}), C_{\text{sk}}) = m$

Known limitations:

- may not terminate: verification is undecidable
- may not be able to conclude (neither proof nor attack)

```
(* Server main process *)
let Server(S_sk: skey, S_cert: certificate,
           crypto: cryptography, mode: chmode,
           check: smode) =
  let S_pk = pk(S_sk) in
  ( (* OPN request and response *)
  (* Receive OPN request *)
  in(c, CHreq: bitstring)
  let CH(OPN_header(ch: chid, =crypto,
         C_cert: certificate, print: thumbprint),
         payload: CH_payload) = CHreq in
  let is_reopen = ch <> NewCh in
  if not(is_reopen) || (reopening_allowed()
     && mode <> None) then
  if mode = None || (
   (*{if fixed}*) print = h(S_pk)
   (*{else}*) (crypto = RSA && print = h(S_pk)) ||
              (crypto = ECC && print = no_print)
   (*{endif}*)) then
  if get_kind(C_cert) = client
     && get_crypto(C_cert) = crypto then
  let C_pk = get_pk(C_cert) in
```

A Formal Analysis with ProVerif



One of the most complex ProVerif formal model

Size of the model

- unfolded process generated: 8.6k LoC of applied π-calculus
- translated into 2.3k initial Horn clauses
- more LoC and Horn clauses than TLS 1.3 in [Bhargavan et al., CCS 2022]

A highly modular and configurable model

- different protocol configurations:
  cryptography, channel security modes, supported user credentials, session security level
- threat model:
  compromise of long-term keys, or channel keys
- allow for simplifications:
  disable channel reopening, or channel switching

$\Longrightarrow$ combinations result in 7056 configurations

Tooling and proof campaigns

An iterative approach, using Python scripts:

```
prove -c "ECC,Sign|Encrypt,reopen,SSec,pwd|cert,switch,lt_leaks"
```

- configurations organized as a lattice for efficient exploration
  - partial order: $C < C' \iff C$ has $\begin{cases} \text{less features, or} \\ \text{a weaker threat model} \end{cases}$ than C'
  - proof on $C \implies$ proof on all $C' < C$
  - inability to prove on $C \implies$ inability to prove on all $C' > C$
- exploration highly parallelized
- iterative exploration by increasing time-outs

A new proof methodology required

Many cases of non-termination due to complex state machine

New methodology combining state-of-the-art features of ProVerif:

- identify the main loops that appear during resolution
- break each loop by redefining selection function guiding proof search
  → inconclusive result (neither proof nor attack)
- iteratively add lemmas (more than 100 for our model)
- different proof hints for lemmas and main queries
  → prove lemmas separately, then assume them as axioms

Soundness is preserved

Practical Results

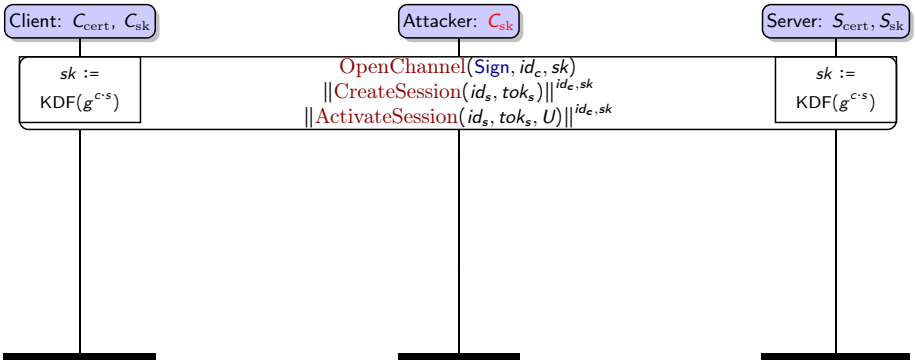
ProVerif results

- new attacks discovered (attack traces provided)
- security proofs for *unaffected* configurations
- inconclusive for the most complex configurations

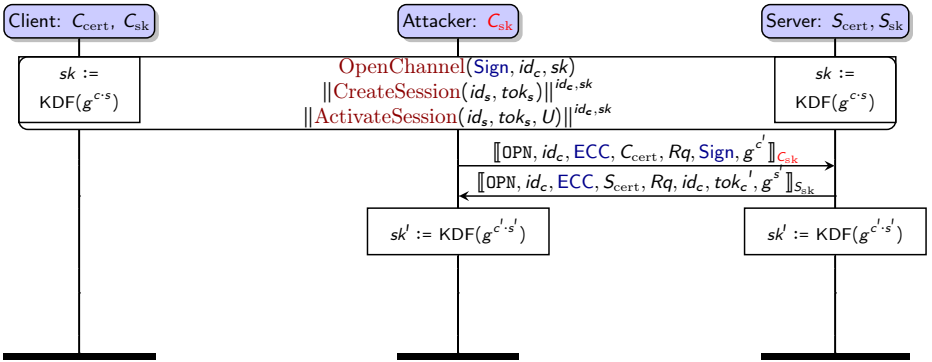
5 new attacks and 3 weaknesses:

- responsibly disclosed through a vulnerability report, tickets and meetings
- upcoming Release Candidate from the OPC Foundation will fix them
- many attacks due to a lack of binding between sessions and secure channels

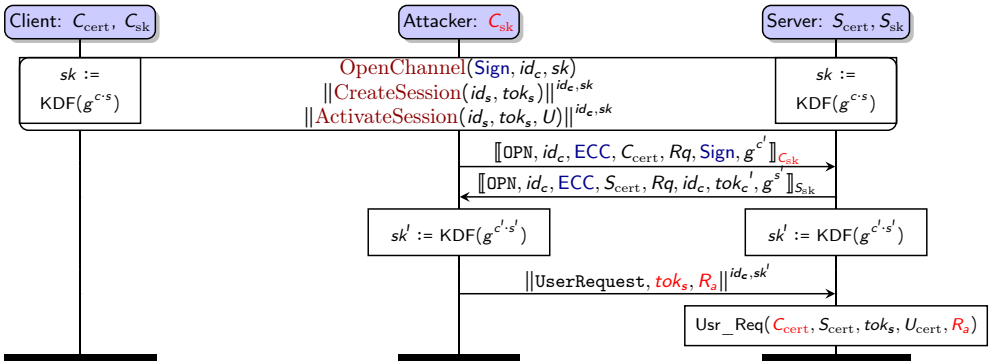
Session hijack by reopening a channel in Sign mode



Session hijack by reopening a channel in Sign mode



Session hijack by reopening a channel in Sign mode



Authentication violation: S receives R_a from (C^a, U) but user U on C did not send it.

- secure channel takeover: *lack of binding between channel keys before and after renewal*
- session hijack: *no ownership proof, tok_s is not secret*

Summary of all reported attacks and weaknesses

Name	Violated properties	Threat	Config.	Impact	Root Causes	Remediations: Fix, Mitigation and Enhancement
Race Condition for User Contexts	$Agr_{\mathrm{CS}}[U]$: S receives a request from C, U_2 but U_1 sent it	∅	∅	Abuse of user rights, wrong logging (user)	• RC1: missing binding btw user and activated session (tok_s)	• Fix1: change tok_s at each user activation • Mit1: limit activations to users with same rights
Client Impersonation in ECC	$Agr_{\mathrm{CS}}[C]$: S receives a request from C^a, U but U sent it on C	$\mathrm{Att}(C^a_{sk})$	ECC + SNoAA	Server confusion (C), wrong logging	• RC2: lack of receiver identity in OPN requests and responses	• Fix2: enforce receiver identity in ECC • Enh2: enforce stronger checks in SNoAA
KCI: User Impersonation	$Agr_{\mathrm{S}}^{0}[U]$: S receives a request from C^a, U but user U on C did not send it	$\mathrm{Att}(C^a_{sk})$ $\mathrm{Att}(S_{sk})$	Cert	User impersonation	• RC3: lack of binding of user's authentication (user's signature) to full context (client identity)	• Fix3: add client's identity in user's signature • Enh3: plan to release some enhanced security versions of the user tokens
Downgrade of Password Secrecy	$Conf_{\mathrm{Pwd}}$: Compromission of user password	$\mathrm{Att}(sk)$	Enc + Pwd	Stealing of user password, full user impersonation	• RC4: relaxed security configuration to avoid what may appear redundant password encryption	• Fix4: use a challenge response mechanism for password authentication or a PAKE protocol • Enh4: dedicated password encryption in Enc
Risk of Signature Oracle	$Conf_{\mathrm{C}}$, $Conf_{\mathrm{Pwd}}$, Agr_{C}: Server impersonation towards (C, U)	∅	ECC + SNoAA	Server impersonation (password stealings)	• RC5: lack of context in signatures	• Fix5: add context and tags to the signature • Enh5: check certificates and nonce lengths
Session Hijack **• Reopen** **(• Switch)**	$Agr_{\mathrm{S}}[U]$: S receives a request from C^a, U but user U on C did not send it	$\mathrm{Att}(C_{sk})$	Sign + Reopen (+ Switch)	User impersonation	• RC6: secure channel takeover at key renewal (Reopen) • RC7: no session ownership proof	• Fix6: linking channel secret keys through renewal • Fix7: keep tok_s secret even is Sign mode, and use it as a MAC computed on each request
KCI: Session and User Confusion	$Agr_{\mathrm{S}}[U]$: S receives a request from C, U but it was sent by a dummy user U^d on C	$\mathrm{Att}(S_{sk})$	Reopen	User impersonation	• RC6: secure channel takeover at key renewal (Reopen) • RC7: no session ownership proof	• Fix6: linking channel secret keys through renewal • Fix7: keep tok_s secret even is Sign mode, and use it as a MAC computed on each request

Summary of all reported attacks and weaknesses

For each reported attack or weakness we provide:

- the violated security property and the attack trace
- the threat assumption, in terms of compromised keys
- the smallest vulnerable configuration
- the practical security impact
- the root cause, often a design flaw (e.g., a lack of authentication secret)
- potential remediations: proposed fixes or mitigations

Conclusion

5 new attacks and 3 weaknesses discovererd:

- Responsible disclosure
- Ongoing discussions with OPC Foundation on fixes

Security proofs for *unaffected* configurations:

- Injective Agreement between Client and Server upon User requests
- Perfect Forward Secrecy of channel keys when using ECC

Future work: proof of properties with the fixes for all configurations

- ProVerif models available
- verification reproducible