Zappa for Correctly Implementing CPSA Analyzed Protocols

John D. Ramsdell
Joshua D. Guttman Ian D. Ketz

HCSS
09 May 2023

MITRE

Problem Statement

Given Security protocol specification that meets its goals

- Eg, verified using analysis tool

Want Protocol implementation meeting specification

- With assurance that it does

MITRE

Problem Statement

Given Security protocol specification that meets its goals

- Eg, verified using analysis tool
- CPSA, for instance

Want Protocol implementation meeting specification

- With assurance that it does
- Zappa compiles the spec
- Coq script validates source/target pairs for simplified core compiler

Same source file for both CPSA and Zappa

MITRE

How does Zappac work?

- CPSA uses the Dolev-Yao adversary model.
- Receptions are destructured using Dolev-Yao elimination rules.
  - Examples: projection from a tuple and decryption.
- Transmissions are synthesized using Dolev-Yao introduction rules.
  - Examples: Tuple construction and encryption.

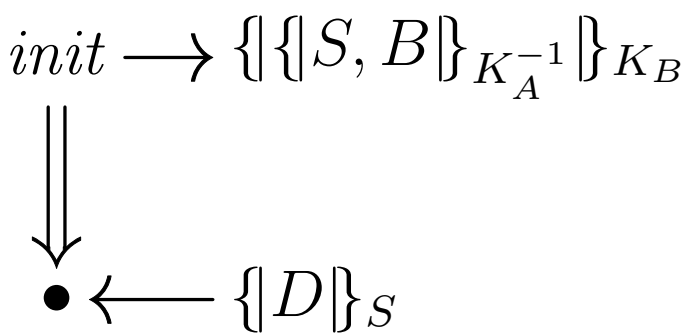
MITRE

Corrected Blanchet Simple Example Protocol

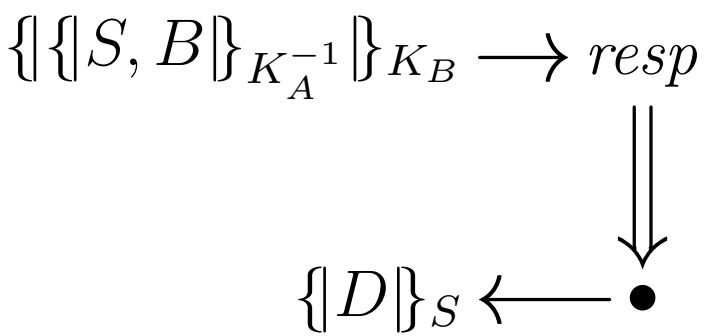
$$A \to B\colon \{|\{|S, B|\}_{K_A^{-1}}|\}_{K_B} \quad [S \text{ generated}]$$
$$B \to A\colon \{|D|\}_S \quad [D \text{ generated}]$$

CPSA Style Roles

Initiator (*init* role)



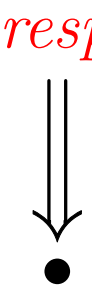
Responder (*resp* role)



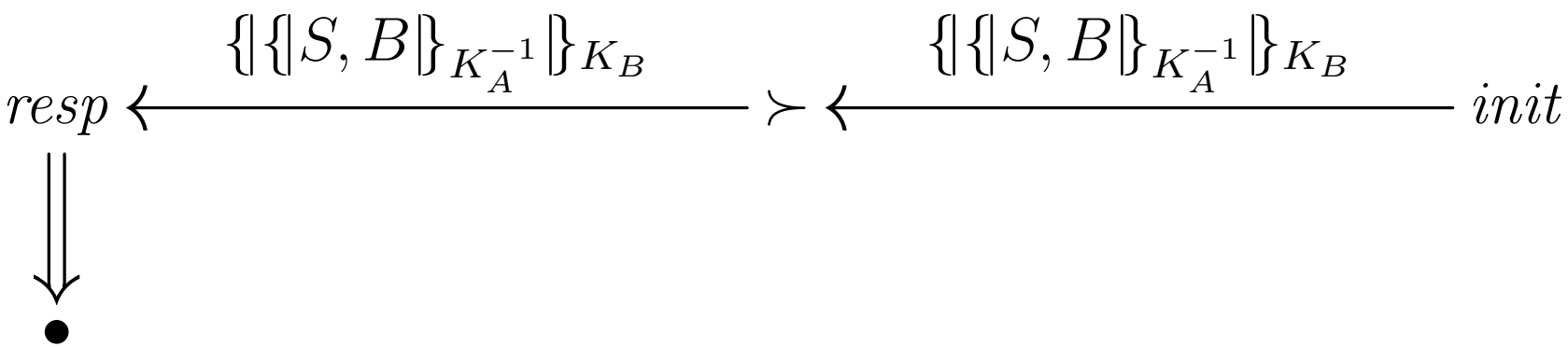
MITRE

CPSA Analysis From the Responder Point-of-View

Query



Analysis



MITRE

CPSA Summary

Cryptographic Protocol Shapes Analyzer

- Query is a protocol and points-of-views.
- Analysis is concise descriptions of allowed behaviors.
- Deduce the goals achieved from the allowed behaviors.

MITRE

Modifications to CPSA to Support Code Generation

Channels: Specify an endpoint for message reception and transmission.

- Channels, as added to CPSA, have a distinct sort.
- Channels cannot be included in messages.

Proc Info: Specify generated procedure parameters and return values.

- All channels must be included in the parameters.

resp:

1. Parameters C, B, K_A, and K_B^{-1}.
2. Receive on C: $\{|\{|S, B|\}_{K_A^{-1}}|\}_{K_B}$.
3. Generate D.
4. Send on C: $\{|D|\}_S$.
5. Return D and S.

MITRE

Zappac Output for the Responder Role

Parameters

1. *Parameters* C, B, K_A, *and* K_B^{-1}.

```
/// Role: resp (blanchet.scm:16:3)
pub fn resp<M, C, Z: Zil<M, C>>(
    z: &Z, v0: &mut C, v1: &M, v2: &M, v3: &M
    ) -> Result<(M, M)> {
  z.chck(b"name", v1)?;
  z.chck(b"akey", v2)?;
  z.chck(b"~akey", v3)?;
```

MITRE

Zappac Output for the Responder Role

Reception

2. *Receive on* C: $\{|\{|S, B|\}_{K_A^{-1}}|\}_{K_B}$.

```
// Recv (blanchet.scm:19:6)
let v4 = z.recv(b"enc", v0)?;
let v5 = z.adec(b"enc", &v4, v3)?;
let v6 = z.vrfy(b"enc", &v5, v2)?;
let v7 = z.proj(b"cat", 2, &v6, 0)?;
z.chck(b"skey", &v7)?;
let v8 = z.proj(b"cat", 2, &v6, 1)?;
z.chck(b"name", &v8)?;
z.same(&v8, v1)?;
```

MITRE

Zappac Output for the Responder Role

Transmission and Return Values

3. *Generate D.*
4. *Send on C: $\{\!|D|\!\}_S$.*
5. *Return D and S.*

```
    // Send (blanchet.scm:20:6)
    let v9 = z.frsh(b"data")?;
    let v10 = z.senc(b"enc", &v9, &v7)?;
    z.send(v0, &v10)?;
    Ok((v9, v7))
}
```

MITRE

Zappa Summary

- Source is a CPSA query.
- Target contains a procedure for each role in the protocol.
- Implement the Zil trait to create an executable.
  - Use the extensive Zappa libraries.
  - Provide your own implementation of the Zil trait.

MITRE

Why is Zappac Output Correct?

- Zappac implements the Roletran algorithm.
  - Roletran's source file is a simplification of Zappac's.
  - Roletran's target file is easily translated into Zappac's.
- A Coq scrip automatically proves Roletran source/target pairs have the same semantics, and therefore achieve the same goals.
  - Script proves:
    - ★ A run of the protocol, implies an execution for its procedure (Liveness).
    - ★ An execution of the proceduce, implies a run of the protocol (Safety).
  - The procedure semantics is specified using a small step semantics.
  - Zil method implementations must honor the small step semantics.

MITRE

Zappa Runtime Systems

- Two distinct runtime system libraries are provided.
- Tools automatically generate parts of a runtime system.
- ASN.1 specified communication formats are supported.
- Zappa system supports protocols with state.

MITRE

Conclusion

Assured Protocol Implementation Scheme

- Use CPSA to ensure your protocol achieves desired goals.
- Use Zappa to ensure the generated code achieves the same goals.
- CPSA and Roletran are available here.
  Roletran paper is here.

MITRE