Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols

Bruno Blanchet
bruno.blanchet@inria.fr

INRIA Paris

August 2017

Case study: ARINC823

- ARINC823 Protocols:
  - Protocols for protection of air-ground messages
  - Two variants: public-key and shared-key
  - Public-key variant implemented by Honeywell
- Constraints:
  - Bandwidth
  - Resistance to failures
  - Flexibility
- Verification:
  - in the symbolic model, using ProVerif
    http://proverif.inria.fr
  - in the computational model, using CryptoVerif
    http://cryptoverif.inria.fr

The public-key protocol (simplified)

Aircraft U — *Ground* V

Secure session initiation

$$m_1{=}(H_1,\text{Init\_REQ},Policy,\mathrm{U},\mathrm{V},Algos,t_\mathrm{U}),\ s_\mathrm{U}{=}\mathrm{SIG}(d_\mathrm{U},(\mathrm{U},\mathrm{V},m_1)) \longrightarrow$$

$Rand_\mathrm{V}$ random

$$Z_{\mathrm{U},\mathrm{V}} = d_\mathrm{V} Q_\mathrm{U}$$

$$X_{\mathrm{U},\mathrm{V}} = \mathrm{HASH}((Algos, t_\mathrm{U}, s_\mathrm{U}, Rand_\mathrm{V}))$$

$$KMAC_{\mathrm{U},\mathrm{V}} = \mathrm{KDF}(Z_{\mathrm{U},\mathrm{V}}; m; \mathrm{concatSI}(1, MAC\_algo, X_{\mathrm{U},\mathrm{V}}, \mathrm{U}, \mathrm{V}, m))$$

$$KENC_{\mathrm{U},\mathrm{V}} = \mathrm{KDF}(Z_{\mathrm{U},\mathrm{V}}; n; \mathrm{concatSI}(2, ENC\_algo, X_{\mathrm{U},\mathrm{V}}, \mathrm{U}, \mathrm{V}, n))$$

$$\longleftarrow m_2{=}(H_2,\text{Init\_RSP+},AlgSel,[cert_\mathrm{V}],t_\mathrm{U},Rand_\mathrm{V}),\ \mathrm{MAC}(KMAC_{\mathrm{U},\mathrm{V}},(\mathrm{V},\mathrm{U},m_2,Algos,t_\mathrm{U},s_\mathrm{U}))$$

$Z_{\mathrm{U},\mathrm{V}} = d_\mathrm{U} Q_\mathrm{V} \ldots$

Application data exchange

$$m{=}(H,Cmd,\mathrm{ENC}(KENC_{\mathrm{U},\mathrm{V}},(0,Count\_DN),Payload),Count\_DN),\mathrm{MAC}(KMAC_{\mathrm{U},\mathrm{V}},(\mathrm{U},\mathrm{V},m)) \longrightarrow$$

$$\longleftarrow m{=}(H,Cmd,\mathrm{ENC}(KENC_{\mathrm{U},\mathrm{V}},(1,Count\_UP),Payload),Count\_UP),\mathrm{MAC}(KMAC_{\mathrm{U},\mathrm{V}},(\mathrm{V},\mathrm{U},m))$$

Optional ground-initiated trigger

Aircraft U *Ground* V

Ground-initiated trigger

$$m_0 = (H_0, \mathsf{Init\_IND}, \mathsf{U}, \mathsf{V}, [cert_{\mathsf{V}}], t_{\mathsf{V}})$$
$$s_{\mathsf{V}} = \mathsf{SIG}(d_{\mathsf{V}}, (\mathsf{V}, \mathsf{U}, m_0))$$
$$\longleftarrow$$

followed by the previous protocol.

Algorithms

- MAC: HMAC-SHA256, truncated to 32, 64, or 128 bits
- Encryption: AES128-CFB128
- Compression: Dynamic Markov Compression or DEFLATE
  When compression is active, the payload is compressed before encryption.

Protection modes

- Several protection modes:
  - BOTH: the message is encrypted, then MACed.
  - AUTH: the message is just MACed
  - NONE: no encryption nor MAC
- The protection mode is chosen based on a policy (policy id sent in the Init_REQ message field *Policy*) and the type of message sent.
- The protection mode of each message is included in the header field *H*.

Protection against replays

The ground entity verifies that the received AMS_Appendix field (*Algos*, t_U, s_U) does not occur in the Init_REQ messages of currently established sessions.

- Attempt to avoid replays of the Init_REQ message.
- Not enough to avoid all such replays.

No replay protection for the ground-initiated trigger.

- When the ground-initiated trigger is replayed, the aircraft stops the current session and starts a new one.

Protection against replays: improved

- For each aircraft U, the ground entity should
  - use a replay cache containing the AMS_Appendix field of the Init_REQ messages received from U with the most recent timestamp t_U, as long as messages with timestamp t_U are accepted.
  - accept a message with timestamp t'_U if and only if $t'_U > t_U$ or ($t'_U = t_U$ and the AMS_Appendix field is not in the replay cache).
- Use the same protection for the ground-initiated trigger.

Security policy

The specification says:

> *the receiving entity verifies all received messages to ensure correct application of the security policy.*

- This is essential for security: never accept an unprotected message when the policy specifies AUTH or BOTH protection.
- This is mentioned in the body of the specification, but not in the guidance in attachment that describes the protocol steps.
  - We recommend adding these policy checks there.
  - Our protocol models obviously include these checks.

Protocol failure

- In case a secure session cannot be established:
  - Messages may be sent in the clear.
  - Incoming clear messages are accepted, independently of the security policy.
- May cause obvious security breaches.
- Mitigation: we recommend reporting the problem
  - to the pilot and
  - to a computer maintenance team on the ground

  in case a secure session cannot be established.
  We recommend mentioning that in the standard.
- We model only the behavior inside secure sessions.
  - With the mitigation above, our security results still hold when no problem is reported to the pilot or to the ground maintenance.

Using the tools

1. Prepare the input file containing
   - the specification of the protocol to study (initial game),
   - the security assumptions on the cryptographic primitives,
   - the security properties to prove.
2. Run ProVerif/CryptoVerif
   - ProVerif: fully automatic
   - CryptoVerif: may need guidance (game transformations to apply)
3. ProVerif answers true, false, or don't know.
   - outputs an attack when false.

   CryptoVerif answers true or don't know.
   - outputs
     - the sequence of games that leads to the proof,
     - a succinct explanation of the transformations performed between games,
     - an upper bound of the probability of success of an attack

     when true.

Assumptions on primitives: ProVerif

- Ideal black-box primitives.
- Diffie-Hellman: $d_U Q_V = d_V Q_U$.

Assumptions on primitives: CryptoVerif

- Common key for signatures and Diffie-Hellman
  $\Rightarrow$ joint security assumption
  - UF-CMA signatures; SUF-CMA for replay protection
  - GDH: Gap Diffie-Hellman
- Two signatures have a negligible probability of being equal.
  - Deterministic signatures would yield attacks (cf shared-key protocol).
- Signatures of cerficates are UF-CMA.
- KDF256 and KDF128 are independent random oracles.

$$\mathsf{KDF256}(k, (X, U, V)) = \mathsf{KDF}(k; 256; \mathsf{concatSI}(1, \text{HMAC-SHA256}, X, U, V, 256))$$
$$\mathsf{KDF128}(k, (X, U, V)) = \mathsf{KDF}(k; 128; \mathsf{concatSI}(2, \text{AES128-CFB128}, X, U, V, 128))$$

- The MACs are SUF-CMA.
- SHA256 is collision-resistant.
- Encryption is IND-CPA provided the IVs are distinct.
  Note: unusual computation of IVs.

Results (1)

We prove:

- Freshness of Init_IND and Init_REQ messages (timestamp)
- No replays of Init_IND and Init_REQ messages
  - With our improved replay protection.
  - Needs a manual argument in ProVerif.
- Entity authentication
  - Non-injective authentication of U to V.
  - Injective authentication of U to V with our improved replay protection.
  - Injective authentication of V to U.

Results (2)

We prove:

- Message authentication, for messages under protection modes AUTH or BOTH.
  - In ProVerif, manual argument to show absence of replays.
- Secrecy of messages, for messages under protection mode BOTH.
  - Unreachability for ProVerif.
  - Indistinguishability for CryptoVerif.
  - Note: The length of the encoded and compressed payload leaks.
- No unknown key share attacks

Results (3)

Properties that do not hold:

- Forward secrecy
  - If d_U or d_V are compromised, the adversary can compute all keys and decrypt all previous messages.
  - Because Diffie-Hellman is done with long-term keys, not with ephemerals.
- Resistance to key compromise impersonation (KCI)
  - If d_U is compromised, the adversary can impersonate V to U and send messages to U as if they came from V.
  - If d_V is compromised and U starts a session with V, the adversary can send messages to V as if they came from U.

These attacks appear only after the compromise of some keys.

Strengthened protocol

- Signed Diffie-Hellman with ephemerals.
  - U and V generate and send ephemerals in the Init_REQ and Init_RSP messages respectively.
  - The ephemerals are used to compute $Z_{U,V}$.
  - The Init_RSP message is signed rather than MAC'ed.
- Satisfies all properties mentioned before, including forward secrecy and resistance to KCI attacks.
  - Diffie-Hellman with ephemerals guarantees forward secrecy.
  - Signatures protect against KCI attacks.

  Proved in CryptoVerif. ProVerif exhausts memory (8 Gb).
- Distinct keys for each primitive
  $\Rightarrow$ no need for a joint security assumption.
- Some increase in cost.

The shared-key protocol (simplified)

Aircraft U *Ground V*

Optional ground-initiated trigger

$$m_0{=}(H_0,\text{Init\_IND},\mathsf{U},\mathsf{V},t_\mathsf{V}),mac_{0,\mathsf{V}}{=}\mathrm{MAC}(K_{\mathsf{U},\mathsf{V}},(\mathsf{V},\mathsf{U},m_0)) \leftarrow$$

Secure session initiation

$$m_1{=}(H_1,\text{Init\_REQ},Policy,\mathsf{U},\mathsf{V},Algos,t_\mathsf{U}),mac_{0,\mathsf{U}}{=}\mathrm{MAC}(K_{\mathsf{U},\mathsf{V}},(\mathsf{U},\mathsf{V},m_1)) \rightarrow$$

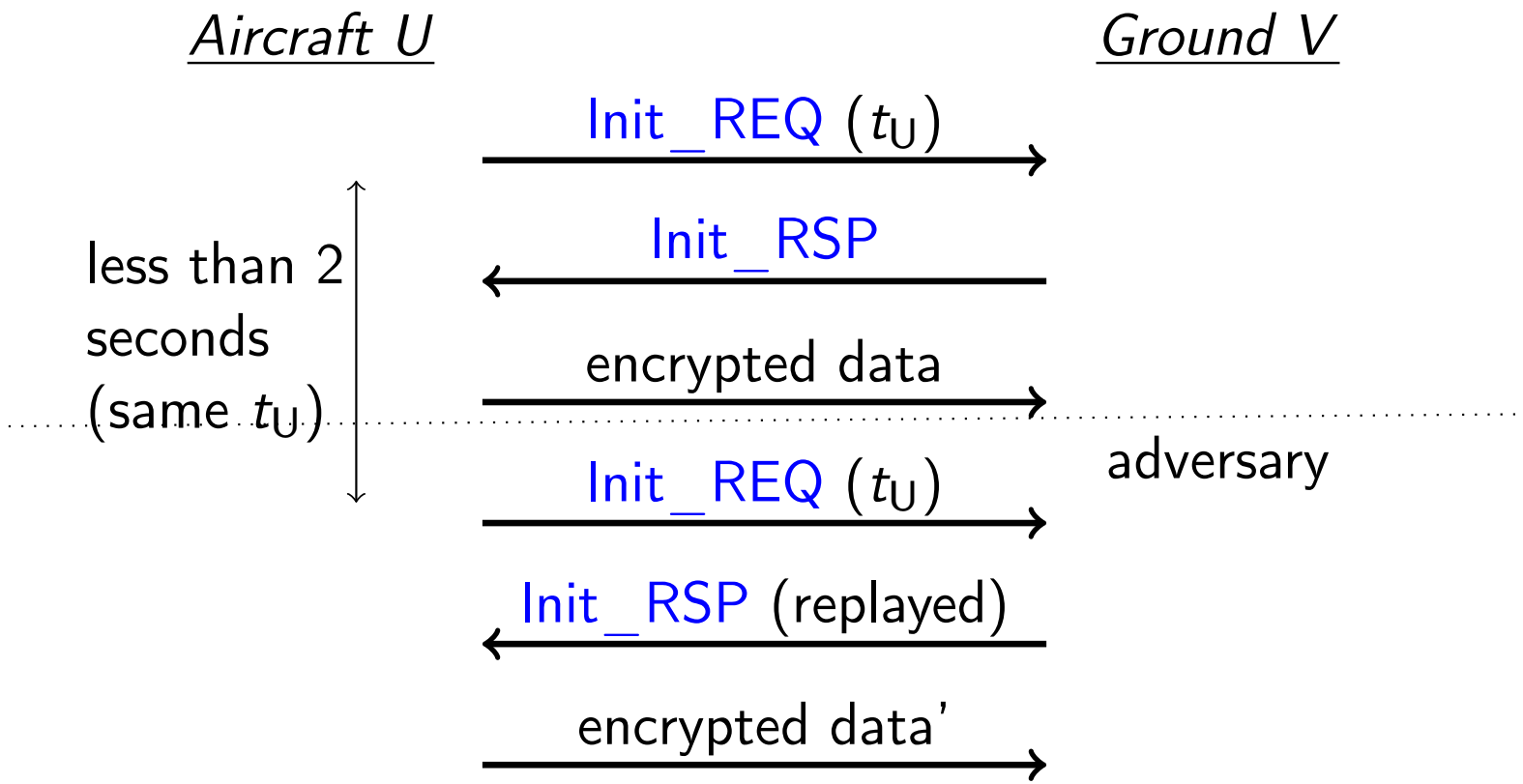
$Rand_\mathsf{V}$ random

$$X_{\mathsf{U},\mathsf{V}} = \mathrm{HASH}((Algos, t_\mathsf{U}, mac_{0,\mathsf{U}}, Rand_\mathsf{V}))$$
$$KMAC_{\mathsf{U},\mathsf{V}} = \mathrm{KDF}(K_{\mathsf{U},\mathsf{V}}; m; \mathrm{concatSI}(1, MAC\_algo, X_{\mathsf{U},\mathsf{V}}, \mathsf{U}, \mathsf{V}, m))$$
$$KENC_{\mathsf{U},\mathsf{V}} = \mathrm{KDF}(K_{\mathsf{U},\mathsf{V}}; n; \mathrm{concatSI}(2, ENC\_algo, X_{\mathsf{U},\mathsf{V}}, \mathsf{U}, \mathsf{V}, n))$$

$$m_2{=}(H_2,\text{Init\_RSP+},AlgSel,[cert_\mathsf{V}],t_\mathsf{U},Rand_\mathsf{V}),\mathrm{MAC}(KMAC_{U,V},(V,U,m_2,Algos,t_\mathsf{U},s_\mathsf{U})) \leftarrow$$

Replay attack that also breaks secrecy (1)

Found because the CryptoVerif proof of secrecy failed.



The two sessions share the same keys.

Replay attack that also breaks secrecy (2)

Consider the n-th message sent by the aircraft, in both sessions.

- Same IV $IV(n)$ in both sessions.
- P_k: k-th block of payload in the first session
  P'_k: k-th block of payload in the second session
  C_k: k-th block of ciphertext in the first session
  C'_k: k-th block of ciphertext in the second session.

$$C_1 = P_1 \oplus \mathrm{AES128}(KENC_{U,V}, IV(n))$$
$$C'_1 = P'_1 \oplus \mathrm{AES128}(KENC_{U,V}, IV(n))$$

so $C_1 \oplus C'_1 = P_1 \oplus P'_1$.
The adversary knows C_1 and C'_1, so it can compute $C_1 \oplus C'_1$ and thus obtain $P_1 \oplus P'_1$: this is an attack against secrecy.

Security results

- Fix: forbid the aircraft from starting two sessions with the same ground entity and same timestamp.
- Then we recover the same security properties as the public-key protocol.
- See the paper for details.

Conclusion

- Verification using ProVerif (symbolic model) and CryptoVerif (computational model).
  - The two approaches are complementary.
- Proved most expected properties,
  in particular secrecy and message authentication.
- Found weaknesses and attacks:
  - clear messages in case of failure,
  - replays,
  - need for a joint security assumption between several primitives,
  - no forward secrecy,
  - KCI attacks.
- Suggested and proved fixes.