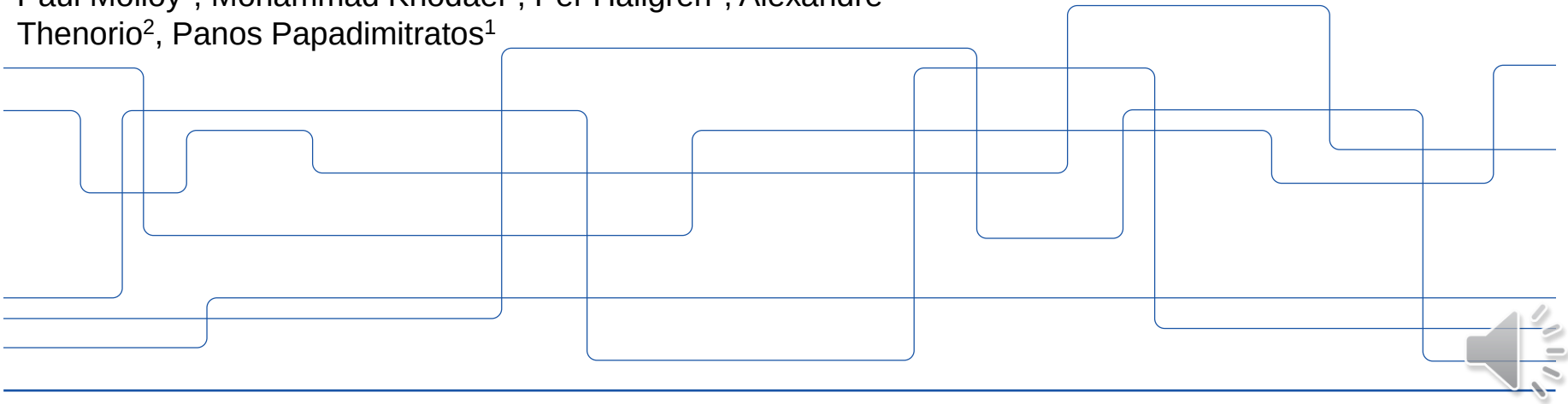
1

einride 2

SecProtobuf

Implicit Message Integrity Provision in Heterogeneous Vehicular Systems

Paul Molloy[1], Mohammad Khodaei[1], Per Hallgren[2], Alexandre Thenorio[2], Panos Papadimitratos[1]



Introduction

Novel Applications of Vehicular Communications:

- Platooning
- Remote Driving

Require strong security Guarantees of:

- Authentication, Integrity and Non-repudiation

IEEE 1609.2 [1] requires these and defines messages

Introduction

Pod (truck)

1. msg.Sign()
2. proto.Marshal(msg)
3. sendUDP(msg)

Remote Drive Station

1. receiveUDP(&bytes)
2. proto.Unmarshal(bytes, &msg)
3. If ok, err := msg.verify();..
4. processData(&msg)

Problem

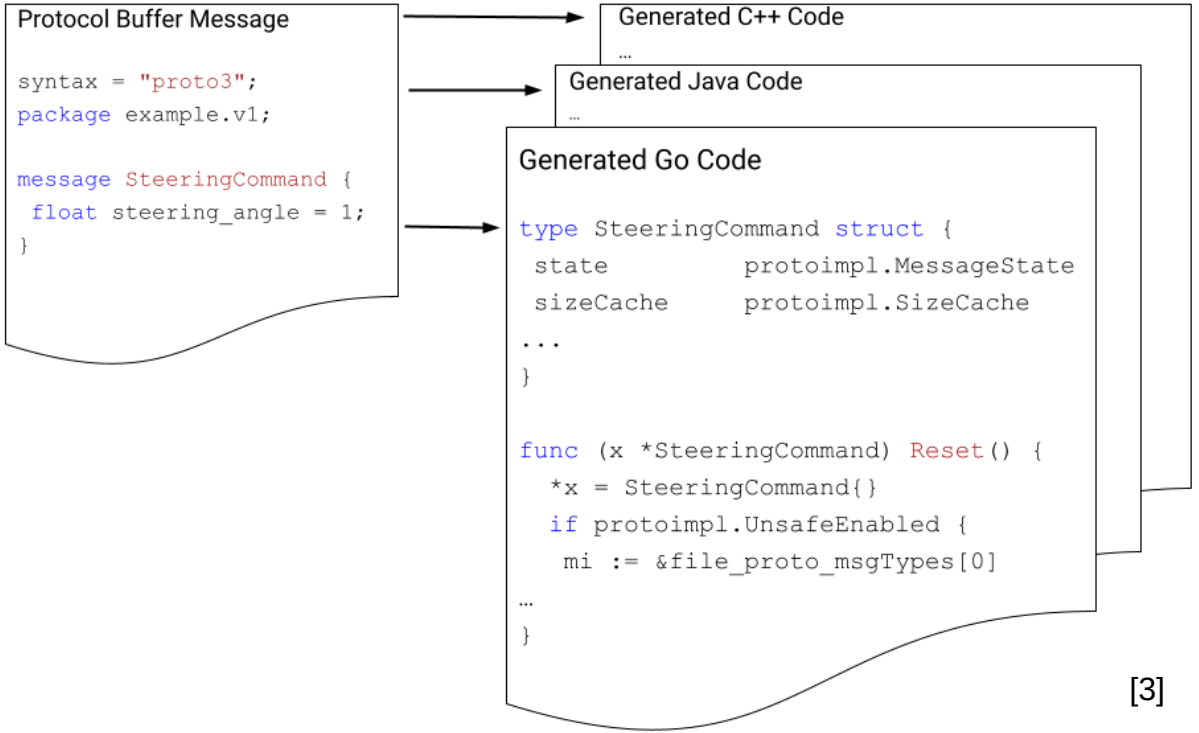
- Re-implementing message security checks for each message type creates additional defects [2]
- Opportunities for key misuse and leaking.
- Developers Forget to check signatures
- Developer time lost linearly due to re-implementation for new message types.

SecProtobuf

Novel security framework:

- Automate the signature generation and validation for VC safety and non-safety data structures
- Mitigate Defect risk
- Speed up deployment of VC use-cases

Methodology

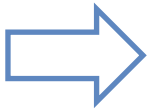


[3]

Methodology

Protocol Buffer with custom option. → Generated Code

```
message SteeringCommand {
  float steering_angle = 1;
  bytes signature = 2
[(integrity.v1.signature) = {
    behaviour: REQUIRED}];
 }
```



```
func(x*SteeringCommand) Sign() error {
  …
 return verification.Sign(x, keyID)
}
func(x*SteeringCommand) Verify() (bool,error) {
  …
 return verification.Validate(x, keyID)
}
```

SecProtobuf Linter

- Check for any proto Message with the message_integrity_signature option set to REQUIRED
- For any msg s.t. proto.Marshal(msg) is called:
  - msg.Sign() is called before it.
- For any msg s.t. proto.Unmarshal(data, msg) is called:
  - msg.Verify() is called before it is accessed again.

```
o/repos/einride/thesis-implicit-message-integrity/internal/integritycheck/testdata/src/a/marshalling.go:39:26: found possible marshalling of integrity proto before signing
o/repos/einride/thesis-implicit-message-integrity/internal/integritycheck/testdata/src/a/marshalling.go:47:26: found possible marshalling of integrity proto before signing
o/repos/einride/thesis-implicit-message-integrity/internal/integritycheck/testdata/src/a/marshalling.go:110:33: found possible unmarshalling of integrity proto without verifying afterwards
```

Results

Run	(De)serialisation	HMAC	ECDSA
Sign	0.000254 ms	0.002165 ms	2.104739 ms
Verify	0.000206 ms	0.001976 ms	0.125568 ms

Conclusion and Outlook

Scaling development of V2X requires tooling for 1609.02 [1] and other standards to:

- Reduce misimplementation security risk
- Speed up developement

Further development of SecProtobuf to add more features, versatility.

SecProtobuf Github: https://github.com/einride/protoc-gen-messageintegrity

References

- [1] “IEEE Standard for Wireless Access in Vehicular Environments(WAVE)–Certificate Management Interfaces for End Entities,”IEEE Std1609.2.1-2020, pp. 1–287, Dec. 2020
- [2] S. McConnell, “Gauging Software Readiness with Defect Tracking,”IEEE Software, vol. 14, no. 3, p. 136, Jun. 1997.
- [3] K. Varda, “Google Protocol Buffers: Google’s data interchange format,”http://code.google.com/apis/protocolbuffers/.

Thanks

- SecProtobuf Github:

https://github.com/einride/protoc-gen-messageintegrity