Università degli Studi di Padova
Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione

Master thesis in Ingegneria Informatica

A receiver centric analysis of the Galileo Open Service Navigation Message Authentication protocol

Supervisor
Prof. Nicola Laurenti

A.A 2017/18

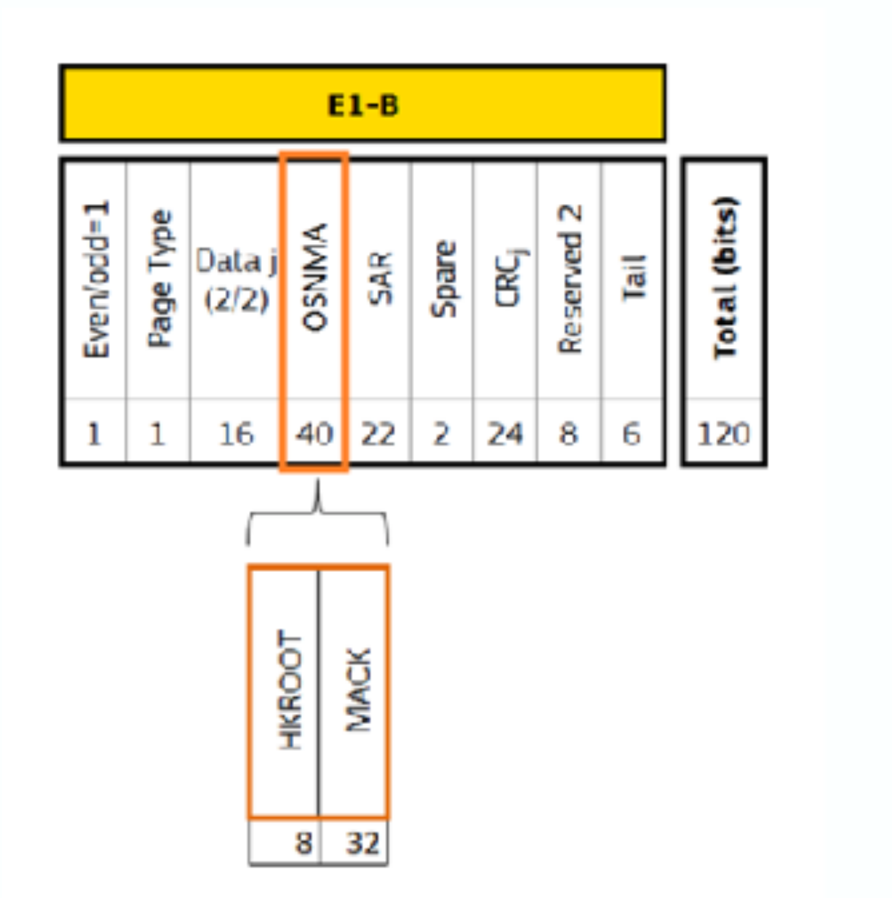
Candidate
Stefano Zanella

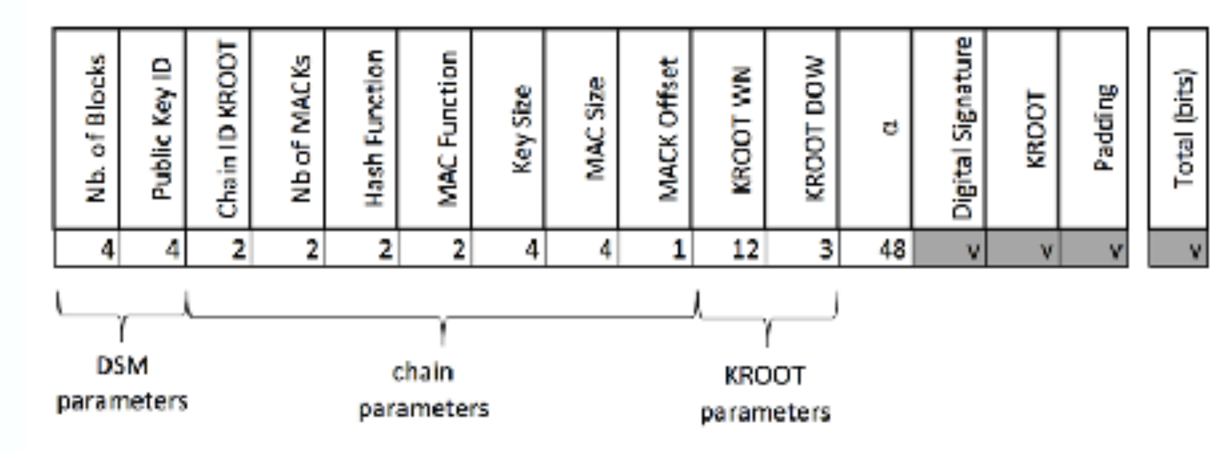
Goals

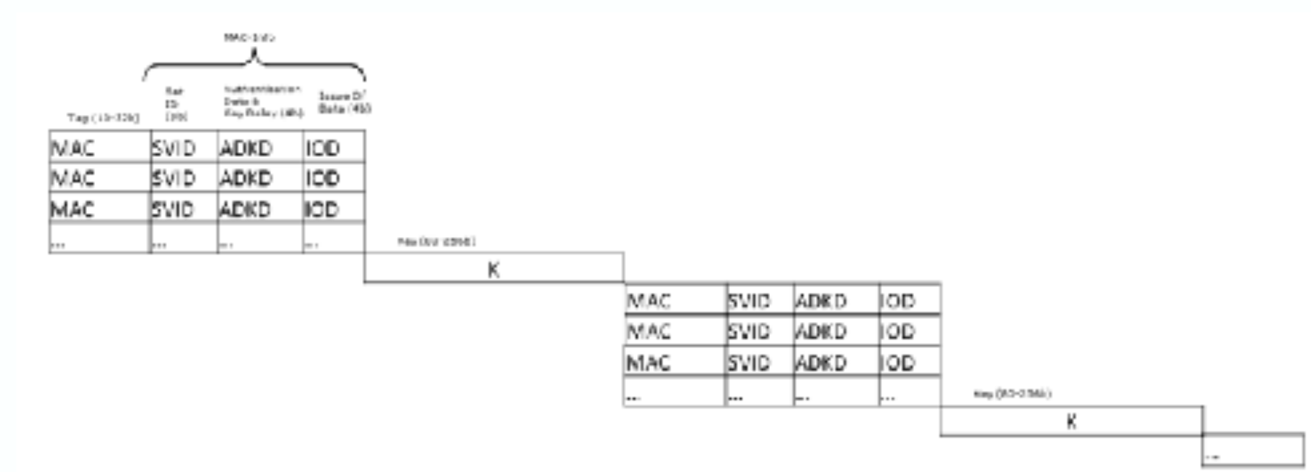
- **Analyse the Galileo OSNMA protocol from the point of view of the receiver**
- **Identify potential issues in the areas of**
  - **Performance**
  - **Robustness**
  - **Complexity**
- **Aid future hardware implementations by providing**
  - **Bounds for best / worst case scenarios**
  - **Guidelines for avoiding security risks**

OSNMA Protocol

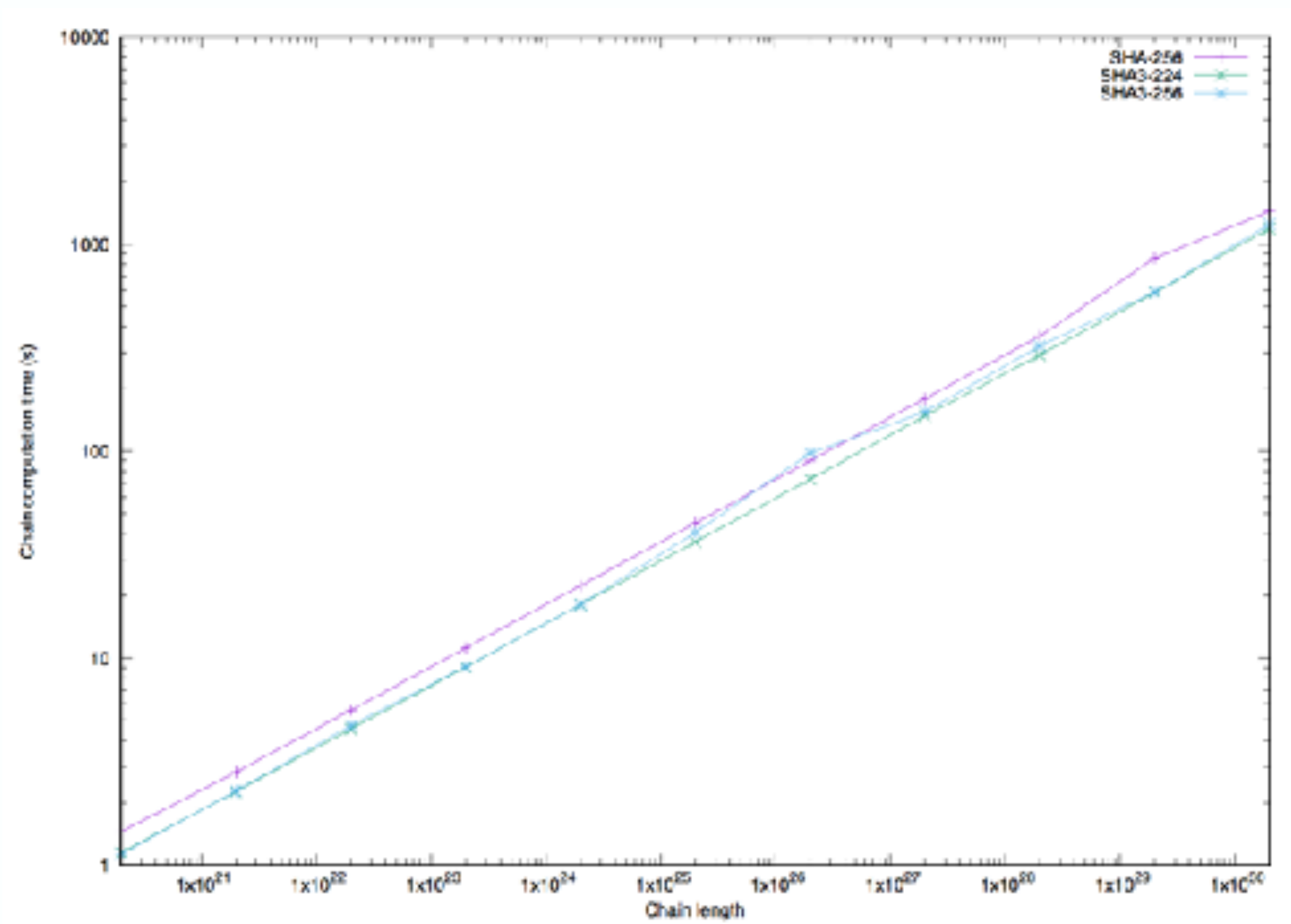
- **Periodically send publicly authenticated root key in DSM-KROOT**
- **MAC for navigation message**
- **Key sent right after in the same MACK section**
- **SLMAC for delayed authentication**







How long does it take to receive and authenticate a key against a root key? (worst case)



$$t_{rj} = \frac{50 \cdot 10^3}{230} t_j \ = 246.30 t_j$$

Chain size	SHA-256	SHA3-224	SHA3-256
2^{20}	355.88	282.14	278.22
2^{21}	691.54	558.24	560.78 ~4.5m
2^{22}	1376.1	1116.2	1159.3
2^{23}	2755.9	2226.1	2239.3
2^{24}	5519.1	4473.1	4475.3
2^{25}	1.1048e4	8956.0	9929.8
2^{26}	2.2127e4	1.7793e4	2.4075e4
2^{27} ~6.7h	4.4199e4	3.6502e4	3.8056e4
2^{28}	8.8459e4	7.1796e4	7.9306e4
2^{29}	2.1065e5	1.4355e5	1.4436e5
2^{30}	3.5531e5	2.9374e5	3.0714e5

Improving bootstrap time with Floating KROOTs

Challenge: calculating the key index

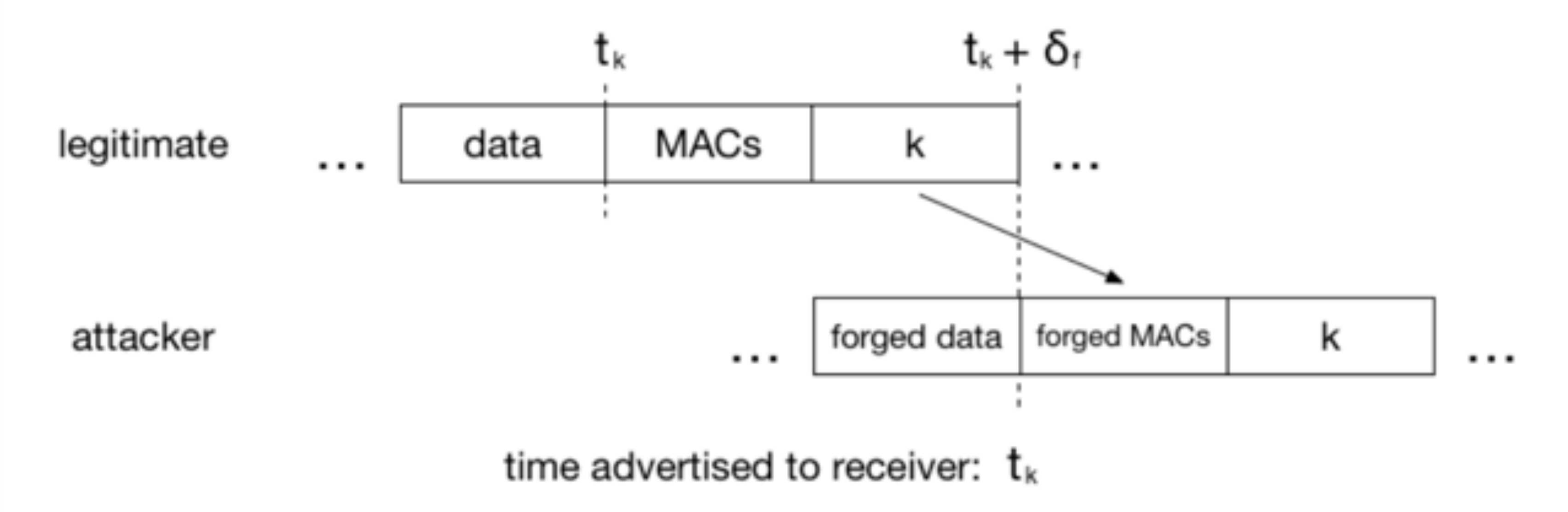
$$\begin{aligned} d &= \frac{(\mathrm{WN}_m - \mathrm{WN}_j)\cdot 604800 + (\mathrm{TOW}_m - \mathrm{DOW}_j \cdot 86400)}{30} n_M \cdot \mathrm{NS} + l \cdot \mathrm{NS} \\ &= \left[\frac{(\mathrm{WN}_m - \mathrm{WN}_j)\cdot 604800 + (\mathrm{TOW}_m - \mathrm{DOW}_j \cdot 86400)}{30} n_M + l\right] \cdot \mathrm{NS} \end{aligned} \qquad (5.8)$$

Time resolution is 1 day, so not all keys can be used as floating KROOTs, but upper bound improves consistently

Distance from KROOT	Time on Intel Core i5 (s)	Est. time on ARM (s)
103680	0.1411	34.753
414720	0.5688	140.10

Attacks against clock synchronisation

Attacker relies on large clock drift to spoof/replay authenticated data



Attacks against clock synchronisation

Solution: use SLMAC
Note: inactivity time needs to stay within threshold

$$t_{th} = \frac{\delta_{max}\,\mathrm{s}}{86400 \cdot d\,\mathrm{ppm}}$$

	Clock precision [ppm]		
	10	1	0.01
80bit key, 30s delay	40.50	405.0	4050
256bit key, 30s delay	53.24	532.4	5324
80bit key, 300s delay	353.0	3530	3.530×10^4
256bit key, 300s delay	365.7	3657	3.657×10^4

Receiver operations

- **Analysis of single core, single thread state machine for data processing**
- **Analysis of memory requirements**
- **Guidelines for processing data at subframe boundary**
- **Exception handling**

Conclusions

- **Adding authentication has a non-negligible impact on receiver complexity**
- **Worst-case scenarios might not fit common use cases**
- **Design of new generation receivers might change to adapt to new requirements (e.g. multi-core, dedicated crypto chip, better clocks)**

Future work

- **Improvements on the timing for a first authenticated position fix**
- **Treatment of error conditions**
- **Reducing complexity on the receiver**
- **Extended analysis of OSNMA energy footprint**

Thank you