VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks

Jo Van Bulck, Jan Tobias Mühlberg and Frank Piessens
jo.vanbulck|jantobias.muehlberg@cs.kuleuven.be
imec-DistriNet, KU Leuven, Celestijnenlaan 200A, B-3001 Belgium

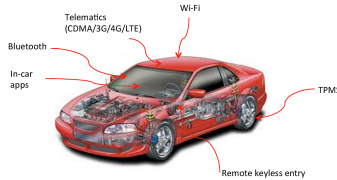
ACSAC, December 2017

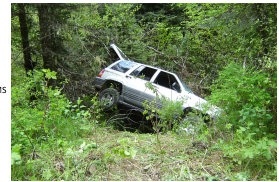
DistriNet

Secure Automotive Computing with VulCAN

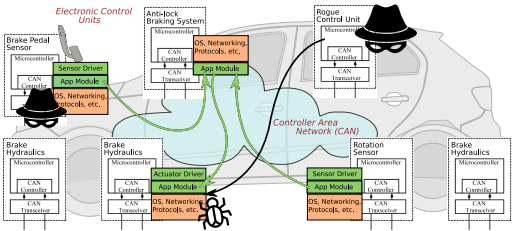
Modern cars can be hacked!

- Network of more than 50 ECUs
- Multiple communication networks
- Remote entry points
- Limited built-in security mechanisms





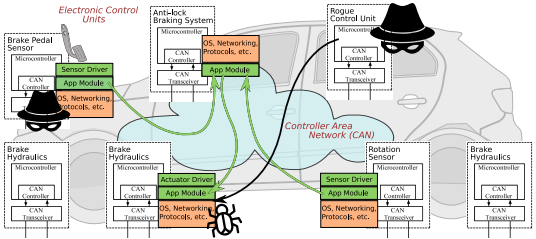
Miller & Valasek, “Remote exploitation of an unaltered passenger vehicle”, 2015



VulCAN brings strong security to automotive computing:

- Message authentication
- Strong software security
- Trusted Computing: software component isolation and cryptography
- Applicable in ICS, IoT, . . .

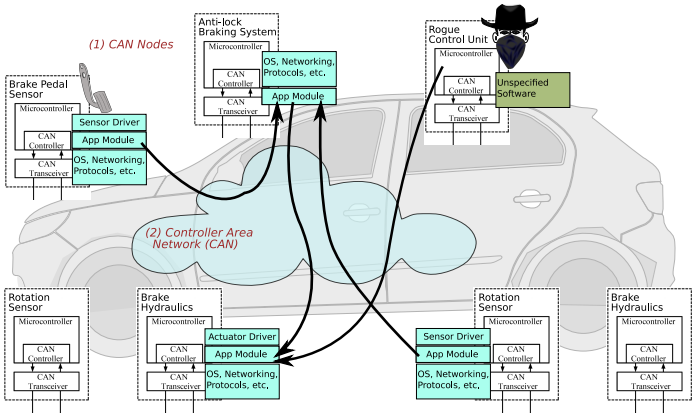
Secure Automotive Computing with VulCAN



VulCAN: Generic design to exploit light-weight trusted computing in CAN-based embedded control networks.

Implementation: based on Sancus [NVBM+17]; we implement, strengthen and evaluate authentication protocols, vatiCAN [NR16] and LeiA [RG16]

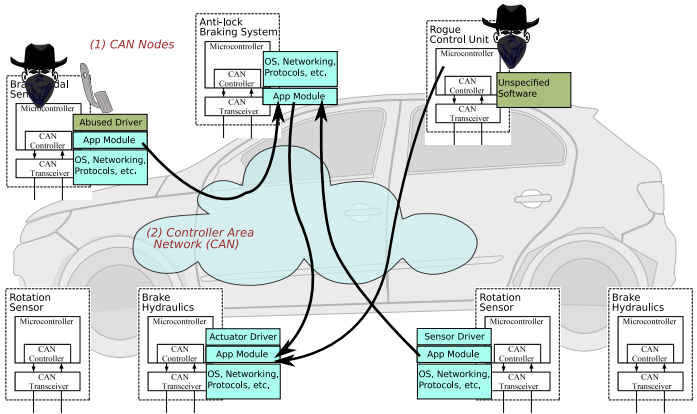
Attacking the CAN



Complex bus system with many ECUs and gateways to other communication systems; no protection against message injection or replay attacks.

→ Message Authentication; specified in AUTOSAR, proposals: vatiCAN, LeiA; no efficient and cost-effective implementations yet

Attacking CAN Message Authentication

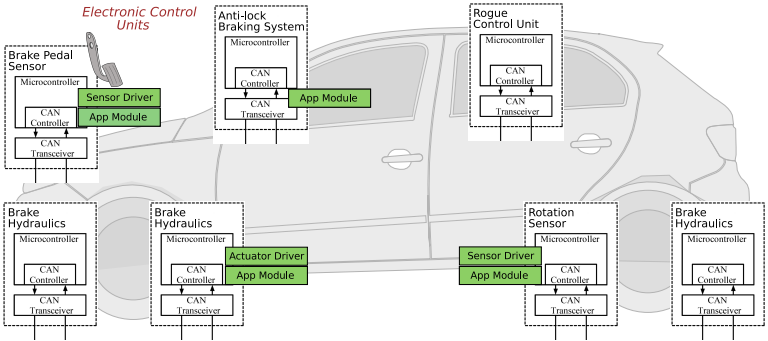


What about Software Security?

Lack of security mechanisms on light-weight ECUs leverages software vulnerabilities: attackers may be able to bypass encryption and authentication.

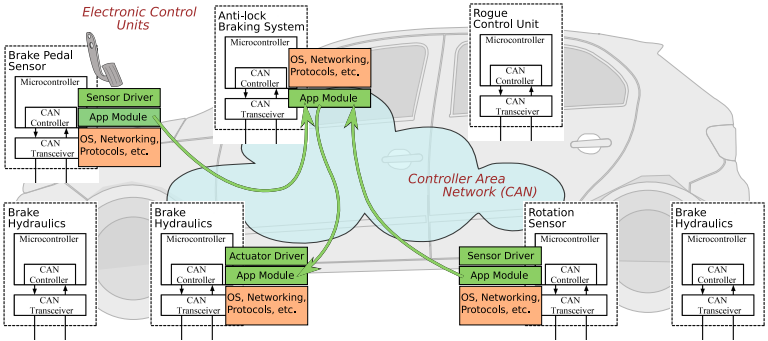
→ Software Component Authentication & Isolation

Overview: Vulcanising Distributed Automotive Applications



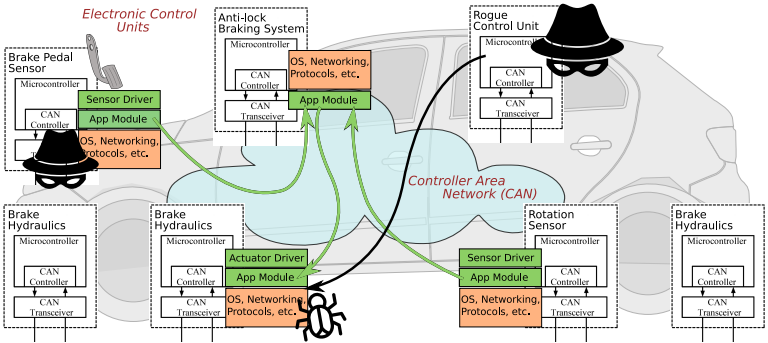
- Critical application components in enclaves: software isolation + attestation

Overview: Vulcanising Distributed Automotive Applications



- Critical application components in enclaves: software isolation + attestation
- Authenticated CAN messages over untrusted system software/network

Overview: Vulcanising Distributed Automotive Applications



- Critical application components in enclaves: software isolation + attestation
- Authenticated CAN messages over untrusted system software/network
- Rogue ECUs, software attackers and errors in untrusted code cannot interfere with security, but may harm availability

Sancus: Strong and Light-Weight Embedded Security [NVBM+17]

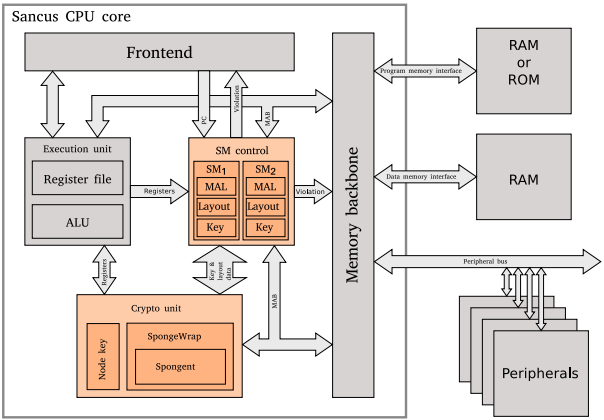
Extends TI's MSP430 with strong security primitives

- Software Component Isolation
- Cryptography & Attestation
- Secure I/O through isolation of MMIO ranges

Efficient

- Modular, $\leq$ 2 kLUTs
- Authentication in μs
- $+$ 6% power consumption

Cryptographic key hierarchy for software attestation



Isolated components are typically very small ($<$ 1kLOC)

Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

Sancus: Strong and Light-Weight Embedded Security [NVBM+17]

Extends TI's MSP430 with strong security primitives

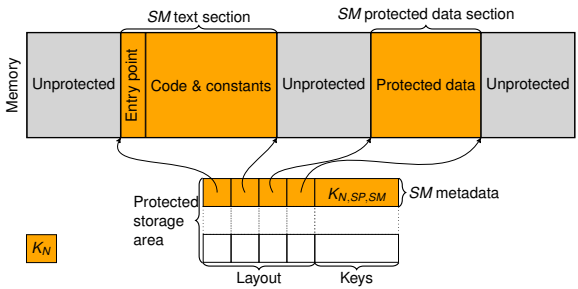
- Software Component Isolation
- Cryptography & Attestation
- Secure I/O through isolation of MMIO ranges

Efficient

- Modular, $\leq$ 2 kLUTs
- Authentication in μs
- + 6% power consumption

Cryptographic key hierarchy for software attestation

N = Node; *SP* = Software Provider / Deployer
SM = protected Software Module



Isolated components are typically very small ($<$ 1kLOC)

Sancus is Open Source: https://distrinet.cs.kuleuven.be/software/sancus/

VulCAN Security Objectives

Protocol requirements

1. Message **authentication**
   ⇒ MAC(id, payload)
2. **Lightweight** cryptography
   ⇒ symmetric keys
3. **Replay** attack resistance
   ⇒ nonces and session keys
4. Backwards **compatibility**
   ⇒ MAC over separate CAN id

vatiCAN [NR16] and LeiA [RG16]

VulCAN Security Objectives

Protocol requirements

1. Message **authentication**
   ⇒ MAC(id, payload)
2. **Lightweight** cryptography
   ⇒ symmetric keys
3. **Replay** attack resistance
   ⇒ nonces and session keys
4. Backwards **compatibility**
   ⇒ MAC over separate CAN id

vatiCAN [NR16] and LeiA [RG16]

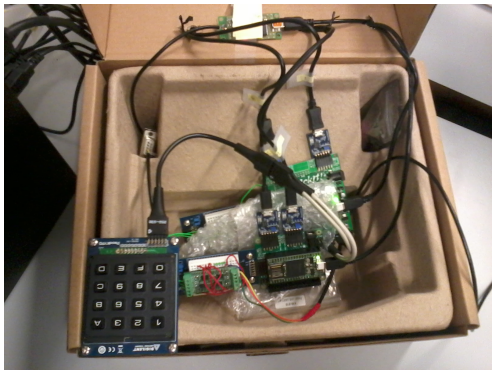
System requirements (with Sancus PMA)

1. **Real-time** compliance
   ⇒ hardware-level crypto
2. Software **isolation**
   ⇒ application + *driver* enclaves
3. Software **attestation**
   ⇒ trusted in-vehicle *attestation server*
4. Dynamic **key**/**ECU update**
   ⇒ via attestation server
5. Secure **legacy ECU** integration
   ⇒ CAN gateway *shielding*

VulCAN Demo Scenario

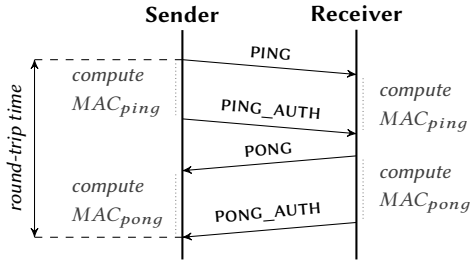
⇒ *distributed authenticated path from keypad to shielded instrument cluster*

⇒ *automotive CAN is challenging – VulCAN is applicable to other domains*



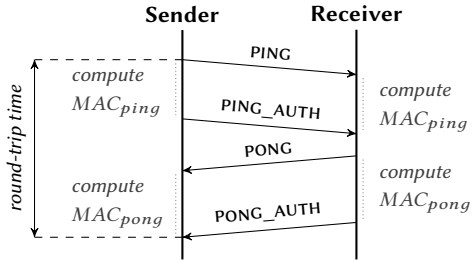
Performance Evaluation: Round-Trip Time Experiment

Scenario	Cycles	Time	Overhead
Legacy	20,250	1.01 ms	–
vatiCAN (extrapolated)	121,992	6.10 ms	502%
Sancus+vatiCAN unprotected	35,236	1.76 ms	74%
Sancus+vatiCAN protected	36,375	1.82 ms	80%
Sancus+LEIA unprotected	42,929	2.15 ms	112%
Sancus+LEIA protected	43,624	2.18 ms	115%



Performance Evaluation: Round-Trip Time Experiment

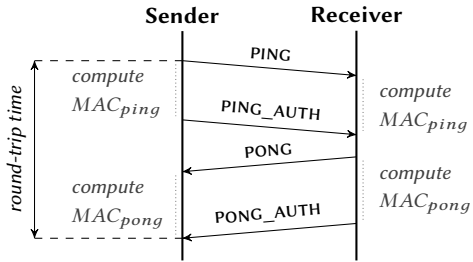
Scenario	Cycles	Time	Overhead
Legacy	20,250	1.01 ms	–
vatiCAN (extrapolated)	121,992	6.10 ms	502%
Sancus+vatiCAN unprotected	35,236	1.76 ms	74%
Sancus+vatiCAN protected	36,375	1.82 ms	80%
Sancus+LEIA unprotected	42,929	2.15 ms	112%
Sancus+LEIA protected	43,624	2.18 ms	115%



- Hardware-level crypto: +400% performance gain ☺

Performance Evaluation: Round-Trip Time Experiment

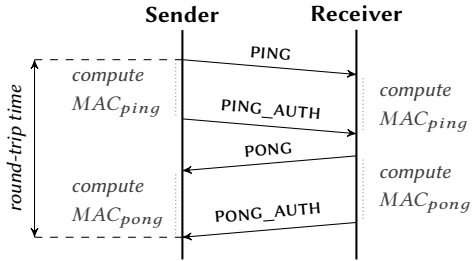
Scenario	Cycles	Time	Overhead
Legacy	20,250	1.01 ms	–
vatiCAN (extrapolated)	121,992	6.10 ms	502%
Sancus+vatiCAN unprotected	35,236	1.76 ms	74%
Sancus+vatiCAN protected	36,375	1.82 ms	80%
Sancus+LEIA unprotected	42,929	2.15 ms	112%
Sancus+LEIA protected	43,624	2.18 ms	115%



- Hardware-level crypto: +400% performance gain ☺
- Modest ~5% performance impact for software isolation [VBNMP15, MNP15]

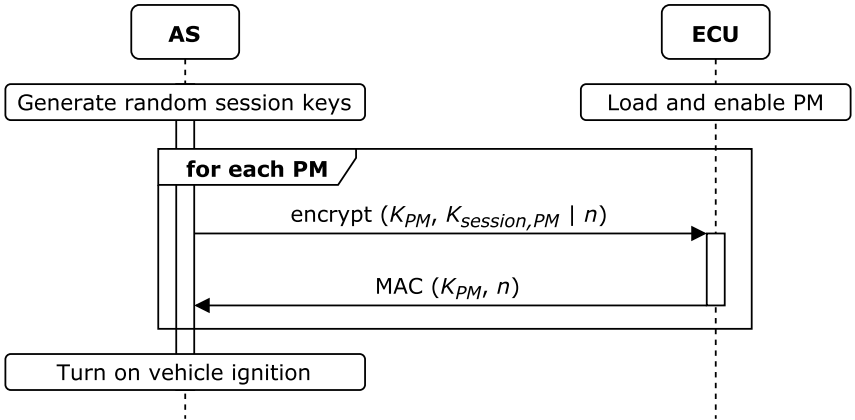
Performance Evaluation: Round-Trip Time Experiment

Scenario	Cycles	Time	Overhead
Legacy	20,250	1.01 ms	–
vatiCAN (extrapolated)	121,992	6.10 ms	502%
Sancus+vatiCAN unprotected	35,236	1.76 ms	74%
Sancus+vatiCAN protected	36,375	1.82 ms	80%
Sancus+LEIA unprotected	42,929	2.15 ms	112%
Sancus+LEIA protected	43,624	2.18 ms	115%



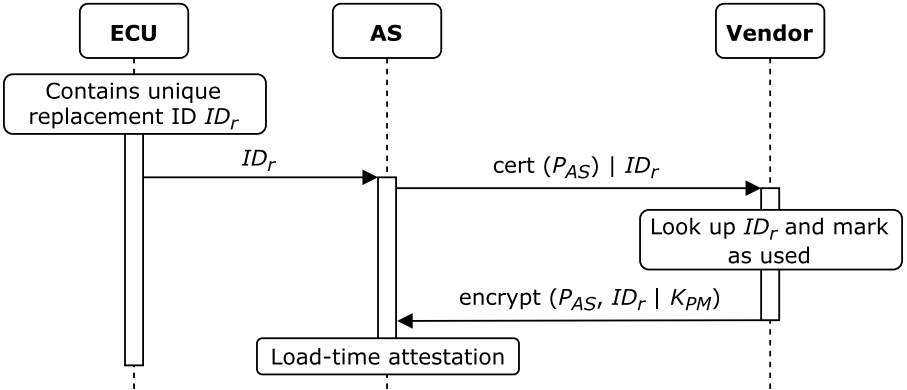
- Hardware-level crypto: +400% performance gain ☺
- Modest ~5% performance impact for software isolation [VBNMP15, MNP15]
- LeiA's extended CAN id usage comes at a cost (SPI-based CAN transceiver)

VulCAN Attestation Server: Boot + Session Key Provisioning



- Challenge-response attestation + encrypted session key distribution
- Preserve motorist safety via secure boot + exclusive vehicle ignition

VulCAN Attestation Server: ECU Replacement



- Untrusted network connection → public key cryptography
- Store software module keys for offline use

Future Work & Research Challenges

Implement vehicle **attestation server:**

- ARM TrustZone vs. Intel SGX: no remote network → *enclave integrity with static root-of-trust* (secure boot)?
- SPONGENT crypto performs bad in software?
- Work-in-progress Rust-SGX [DDL+17] memory-safe implementation

Availability and real-time guarantees on compromised ECUs

→ protected scheduler [VBNMP16]; network availability out of scope (or partition via gateway)

Formalise **authentic execution** [NMP17] guarantees

→ unified framework to interact/deploy enclaves on heterogeneous networked platforms?

VulCAN Security Objectives & Summary

MESSAGE AUTHENTICATION

Receivers get a strong guarantee that a message with specified ID and payload was indeed sent by a trusted sender component.

LIGHTWEIGHT CRYPTOGRAPHY

The use of public key cryptography is ruled out, for typical ECUs are severely constrained in computational power.

REPLAY ATTACK RESISTANCE

The authentication scheme is immune to replay attacks, even when large amounts of traffic were captured.

BACKWARDS COMPATIBILITY

VulCAN remains compatible with off-the-shelf CAN transceiver chips. Legacy applications continue to function.

REAL-TIME COMPLIANCE

VulCAN preserves real-time deadlines for safety-critical functionality such as brakes and steering while not under attack.

COMPONENT ISOLATION

Message processing algorithms and I/O operations are protected against an attacker with *arbitrary* untrusted code execution .

COMPONENT ATTESTATION

Critical software components are guaranteed to be isolated on specific ECUs without having been tampered with.

DYNAMIC KEY UPDATE

VulCAN supports secure key provisioning at runtime, and allows ECUs to be replaced by a distrusted automobile repair shop.

LEGACY ECU INTEGRATION

VulCAN makes it possible to transparently *shield* unmodified legacy ECUs as a transition measure.

Thank you! Questions?

https://distrinet.cs.kuleuven.be/software/vulcan/

https://github.com/sancus-pma/vulcan/

References I

AUTOSAR Specification 4.3.
Specification of module secure onboard communication.
https://www.autosar.org/standards/classic-platform/release-43/software-architecture/safety-and-security/, 2016.

Y. Ding, R. Duan, L. Li, Y. Cheng, Y. Zhang, T. Chen, T. Wei, and H. Wang.
Poster: Rust sgx sdk: Towards memory safety in intel sgx enclave, 10 2017.

J. T. Mühlberg, J. Noorman, and F. Piessens.
Lightweight and flexible trust assessment modules for the Internet of Things.
In *ESORICS '15*, vol. 9326 of *LNCS*, pp. 503–520. Springer, 2015.

C. Miller and C. Valasek.
Remote exploitation of an unaltered passenger vehicle.
Black Hat USA, 2015.

J. Noorman, J. T. Mühlberg, and F. Piessens.
Authentic execution of distributed event-driven applications with a small TCB.
In *STM '17*, vol. 10547 of *LNCS*, pp. 55–71, Heidelberg, 2017. Springer.

S. Nürnberger and C. Rossow.
– vatiCAN – Vetted, authenticated CAN bus.
In *Cryptographic Hardware and Embedded Systems – CHES '16: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pp. 106–124, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

References II

J. Noorman, J. Van Bulck, J. T. Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, J. Götzfried, T. Müller, and F. Freiling.
Sancus 2.0: A low-cost security architecture for IoT devices.
ACM Transactions on Privacy and Security (TOPS), 20:7:1–7:33, 2017.

A.-I. Radu and F. D. Garcia.
LeiA: A lightweight authentication protocol for CAN.
In *Computer Security – ESORICS '16: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II*, pp. 283–300, Cham, 2016. Springer International Publishing.

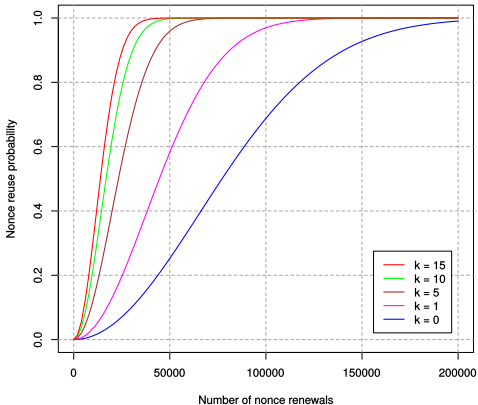
J. Van Bulck, J. Noorman, J. T. Mühlberg, and F. Piessens.
Secure resource sharing for embedded protected module architectures.
In *WISTP '15*, vol. 9311 of *LNCS*, pp. 71–87. Springer, 2015.

J. Van Bulck, J. Noorman, J. T. Mühlberg, and F. Piessens.
Towards availability and real-time guarantees for protected module architectures.
In *MASS '16, MODULARITY Companion 2016*, pp. 146–151, New York, 2016. ACM.

Appendix: vatiCAN Nonce Generator Birthday Attack

(128 bit key + 32 bit nonce + msg) $\rightarrow$ *64 bit MAC*

packet loss $\Rightarrow$ **reset all nonces to random** global value every 50 ms



k-near nonce collision probability:

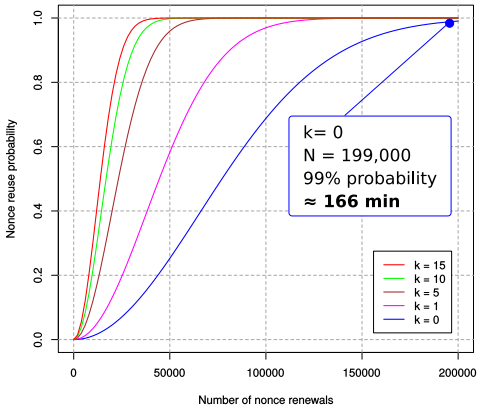
$$p(n,k,d) \approx 1 - \frac{(d-nk-1)!}{d^{n-1}(d-n(k+1))!}$$

with $d = 2^{32}$ (vatiCAN nonce size)

Appendix: vatiCAN Nonce Generator Birthday Attack

(128 bit key + 32 bit nonce + msg) $\rightarrow$ *64 bit MAC*

packet loss $\Rightarrow$ **reset all nonces to random** global value every 50 ms



k-near nonce collision probability:

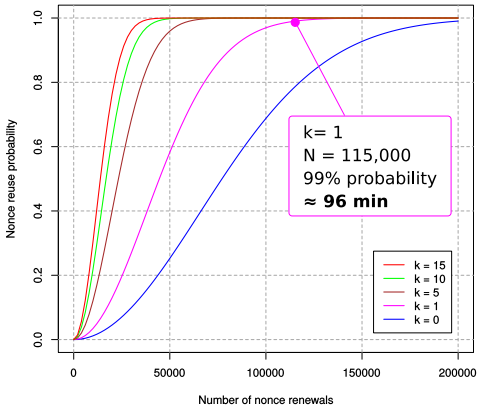
$$p(n,k,d) \approx 1 - \frac{(d-nk-1)!}{d^{n-1}(d-n(k+1))!}$$

with $d = 2^{32}$ (vatiCAN nonce size)

Appendix: vatiCAN Nonce Generator Birthday Attack

(128 bit key + 32 bit nonce + msg) $\rightarrow$ *64 bit MAC*

packet loss $\Rightarrow$ **reset all nonces to random** global value every 50 ms



k-near nonce collision probability:

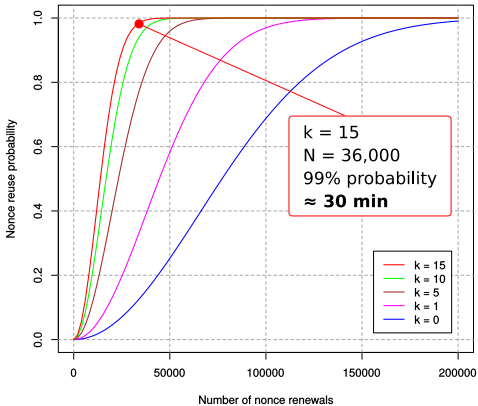
$$p(n,k,d) \approx 1 - \frac{(d-nk-1)!}{d^{n-1}(d-n(k+1))!}$$

with $d = 2^{32}$ (vatiCAN nonce size)

Appendix: vatiCAN Nonce Generator Birthday Attack

(128 bit key + 32 bit nonce + msg) $\rightarrow$ *64 bit MAC*

packet loss $\Rightarrow$ **reset all nonces to random** global value every 50 ms



k-near nonce collision probability:

$$p(n,k,d) \approx 1 - \frac{(d-nk-1)!}{d^{n-1}(d-n(k+1))!}$$

with $d = 2^{32}$ (vatiCAN nonce size)