µDTNSec: A Security Layer for Disruption-Tolerant Networks on Microcontrollers

Dominik Schürmann, Georg von Zengen, Marvin Priedigkeit and Lars Wolf
Institute of Operating Systems and Computer Networks, TU Braunschweig
Email: {schuermann, vonzengen, priedigk, wolf}@ibr.cs.tu-bs.de

***Abstract*—We introduce µDTNSec, the first fully-implemented security layer for Delay/Disruption-Tolerant Networks (DTN) on microcontrollers. It provides protection against eavesdropping and Man-in-the-Middle attacks that are especially easy in these networks. Following the Store-Carry-Forward principle of DTNs, an attacker can simply place itself on the route between source and destination. Our design consists of asymmetric encryption and signatures with Elliptic Curve Cryptography and hardware-backed symmetric encryption with the Advanced Encryption Standard. µDTNSec has been fully implemented as an extension to µDTN on Contiki OS and is based on the Bundle Protocol specification. Our performance evaluation shows that the choice of the curve (secp128r1, secp192r1, secp256r1) dominates the influence of the payload size. We also provide energy measurements for all operations to show the feasibility of our security layer on energy-constrained devices.**

I. Introduction

In recent years Internet of Things (IoT) technology emerged into more and more fields where sensitive data is handled. Namely industrial applications like process monitoring where production data needs to be protected from being accessed unauthorized. Another situation is in the field of intelligent transportation systems, where control data must not be manipulated by an attacker [1]. The most challenging fields are healthcare applications. IoT technology has to face several challenges at once: The first challenge is the mobility of nodes. Devices are mounted to patients, therefore they have to cope with mobility and thus disruption of connections. To ensure a curtain level of reliability in terms of data delivery, these networks have to cope with the mobility of their nodes. Also, the security layer of such a network needs to be tolerant against network disruption. Considering frequent connection losses and short connection periods, key exchange mechanisms like Diffie Hellman might not be able to exchange a key prior every connection.

The second challenge is the variety of communication partners and their different access rights to the data. For example, an X-ray unit might only need an identifier of the patient but a doctor needs the full history of the patient.

To keep the costs of these systems low, direct connections as well as multihop connections are needed. Introducing multihop and different access rights in one network makes traditional IoT security concepts not applicable. For instance in a network using IEEE 802.15.4, a key can be specified that is used by the MAC sublayer for full link layer AES encryption. Thus, every device in the network needs to have the same key to be able to read routing information inside the packets. This means, every device is able to read all data it forwards. Another problem with this encryption is that if one device is hijacked and an attacker was able to read out its key, the attacker is able to read all data transferred in the whole network. This is a severe issue when taking into account that the devices will be handed out to patients for several days or weeks.

In this paper we present a system that overcomes these challenges. To overcome the challenge of disruptions in the connections, we propose to use a lightweight implementation of the bundle protocol [2] called µDTN [3]. By utilizing public key cryptography we solve the security part of the disruption challenge. Another advantage of public key cryptography is the ability to ensure authentication.

The most profound advantage of public key cryptography is the robustness against Man-in-the-Middle attacks. By using a light weight design of the bundle security protocol we enable IoT devices to use all these advantages.

II. Related Work

A possible protocol for Delay/Disruption-Tolerant Networks (DTNs) has been standardized as the Bundle Protocol (BP) in RFC 5050 [2]. An extension with security block types, security processing rules, and ciphersuites is specified in RFC 6257 [4]. Ciphersuites are based upon the cryptographic primitives of RSA with SHA-256 for digital signatures, Keyed-Hash Message Authentication Code (HMAC) for data authenticity, RSA for encrypting symmetric session keys, and Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) for authenticated data encryption. While most desktop implementations, such as IBR-DTN [5] and DTN2 [6], support this extension, implementations for embedded devices lack security features. There are currently three actively maintained implementations of DTN for microcontrollers:

µDTN[1] [3] is our implementation of the BP. It was implemented to meet the hardware and energy constrains of wireless sensor nodes. To meet these we use the compressed bundle header encoding [7]. Originally, µDTN was developed for 8-Bit micro controllers running Contiki OS[2]. By transmitting bundles directly in IEEE 802.15.4 frames we reduced the overhead to a minimum, as shown by Pöttner et al. [8]. To the best of our knowledge it was the first RFC 5050-compatible DTN implementation for wireless sensor nodes.

[1] https://www.ibr.cs.tu-bs.de/projects/mudtn/
[2] http://contiki-os.org

Recently, it was ported to run with FreeRTOS[3] on STM32F4 microcontrollers under the name miniDTN [9]. Both, µDTN and miniDTN, share the same architecture and most of the code. Therefore, in the remainder we handle them as one.

µPCN[4] [10] is a microcontroller-compatible BP implementation that focuses on CubeSat applications. Therefore, the authors consider a routing scenario with a single hop from the satellite to a ground station and multiple nodes behind the ground station. In satellite applications, not every node is reachable from all ground stations and malicious ground station might announce neighbours they can not reach. To tackle these special challenges, µPCN's decision whether to transmit a bundle to the ground station is based on two factors: the reliability and the trustworthiness. Reliability is the number of successful contacts minus the number of unsuccessful ones. Trustworthiness is the probability that a transmitted bundle reaches its destination. With these factors µPCN implements routing security but other security and authentication aspects are left to future work.

DTNLite [11] is an implementation of the concepts of DTN for TinyOS[5]. Instead of implementing the BP, it relies on existing routing protocols and therefore forms an overlay network. The typical DTN Store-Carry-Forward mechanism is implemented by storing data in non-volatile memory on nodes and transporting it towards a single sink over multiple hops. The authors do not consider encryption or authentication in their work.

While these DTNs do not provide confidentiality and authenticity, security on microcontrollers has been studied in other contexts. Because asymmetric RSA operations do not scale linearly with their key size [12] and secure keys must be at least 3072 bit [13], RSA is not practical for embedded systems. Thus, for resource constrained networks, encryption is often implemented by pre-deploying symmetric keys. To automate this process, plenty of key pre-distribution protocols have been designed. However, symmetric key distribution does not scale for huge networks as envisioned for the IoT [14]. Thus, more performant asymmetric operations, such as Elliptic Curve Cryptography (ECC), have been evaluated for ATmega128 and CC1010 microcontrollers by Gura et al. [12]. With NanoECC [15], a fast implementation for TinyOS has been written in C/nesC with assembly optimizations.

Another approach has been developed by Oliveira et al. [16], [17]. Their security layer is based on Identity Based Cryptography (IBC) to encrypt messages by deriving public keys from the nodes' IDs. Their last implementation TinyPBC is based on cryptographic pairings [17]. For this, the authors developed a cryptographic library for microcontrollers called RELIC [18]. It has been shown that RELIC provides the fastest operations but with higher RAM usage in comparison to TinyECC and Wiselib [19]. Later, several improvements have been made to RELIC, e.g., performance improvements on ARM Cortex-M0+ [20]. Due to the costs of cryptographic co-processors, asymmetric operations are still implemented in software. In contrast, most modern radio chips support symmetric algorithms, such as the AES in hardware. In this paper, we leverage the low-level operations of AES provided by the AT86RF23X chip [21]. Its Random Number Generator (RNG) has been evaluated previously during the design of RAIM [22].

Besides cryptographic primitives, full security layers have been proposed and implemented for TinyOS and Contiki. TinySec [23] provides modes for authentication-only (TinySec-Auth) and fully authenticated encryption (TinySec-AE). Here, the Skipjack cipher is used in Cipher Block Chaining (CBC) mode. TinyKey [24] provides key management for TinySec, but introduces no algorithm changes. A successor of TinySec called MiniSec [25] still uses Skipjack, but in Offset Codebook (OCB) mode. All these use Skipjack instead of a more widely used and standardized block cipher, such as AES. Asymmetric ECC operations have been introduced by Liu et al. for key exchange and digital signatures [26].

For Contiki, ContikiSec [27] has been designed and implemented. It provides modes for encryption-only (ContikiSec-Enc), authentication-only with CMAC (ContikiSec-Auth), and a full authenticated encryption mode by using AES in OCB mode (ContikiSec-AE). All modes require a single network-wide key that is pre-deployed on all devices. Optionally, by using other key exchange methods, a session key can be plugged in. ContikiSec uses standardized and well researched cryptographic primitives.

For the purpose of transmitting sensor data in Body Area Networks, One Time Pads (OTPs) have been proposed by Büsching and Wolf [28]. OTPs are deployed during the time where sensors are recharged and used for encryption during the time the sensors are worn. While the authors use µDTN for transmitting sensor data, their security methods have not been integrated into the protocol.

To the best of our knowledge, there is no implementation of a DTN for embedded systems that provides confidentiality and authenticity. In contrast to existing work, we provide a fully integrated solution for resource-constrained DTNs with software-based ECC using the NanoECC [15] and AVR-Crypto-Lib [29] libraries and hardware-based AES using the AT86RF23X radio.

III. µDTNSec

Due to DTN's architecture, where bundles are carried and forwarded by many hops on a route, Man-in-the-Middle attacks can be done easily. Thus, µDTNSec provides end-to-end security between source and destination. It has been designed as a security layer for µDTN and provides two main modes of operation: Signature-only mode for authenticity and Sign-then-Encrypt mode for combined authenticity and confidentiality. A short reference specification is given in Table I. In the following, we discuss µDTNSec's design and modes in detail and provide a threat model.

[3]http://freertos.org
[4]https://upcn.eu
[5]https://github.com/tinyos

TABLE I: µDTNSec Specification: Supported modes with their corresponding BP block, BP ciphersuite and design details

Mode	Block	Ciphersuite	Details
Signature	PIB	PIB-ECDSA-SHA256	Digital signature generation and verification using ECDSA with SHA-256.
Sign-then-Encrypt	PIB +PCB	PCB-ECDH-SHA256-AES128-PIB-ECDSA-SHA256	After signature generation, a shared secret is calculated by ECDH on valid public keys. A session key is derived using the ANSI-X9.63-KDF and used for hardware-backed AES-128 encryption in CBC mode. The payload is encrypted in-place while the PIB is encrypted separately.

Supported SECG [30] curves: secp128r1, secp192r1, secp256r1

A. Threat Model

Before discussing the design of µDTNSec, we first specify a clear threat model. This allows us to derive a set of security features.

Eavesdropping: The typical threat for all network communications is passive eavesdropping. In a wireless network without payload encryption an attacker can eavesdrop on the wireless channel and read transmitted data.

Man-in-the-Middle: In networks with peer-to-peer encryption, but no end-to-end encryption, an attacker could place herself into the route between source and destination to read bundles intended for the victim. This attack can be extended by announcing perfect routing costs to gather a large amount of network bundles. In comparison to passive eavesdropping, a Man-in-the-Middle attack is active and requires a more sophisticated attacker with network knowledge.

Data Modification: In a variation of the Man-in-the-Middle attack, the attacker not just reads but modifies bundles in transit. Every node participating in Store-Carry-Forward routing can do this.

Impersonation: For this attack, the attacker impersonates the victim to receive its bundles or to transmit malicious bundles in the victim's name. To prevent the victim from receiving them, additional routing attacks can be executed, such as blackhole attacks, where packets are dropped instead of forwarded.

B. Security Features

Our threat model covers the most important threats posed by *outside attackers*. µDTNSec's full Sign-then-Encrypt mode provides confidentiality, node authenticity/non-reputability, and data integrity.

Node authenticity provides access control to the network and thus also protects against outside attackers trying to execute routing attacks. Still, inside attackers, i.e., nodes that are already part of the network, are considered out of scope for µDTNSec. For the typical use cases of sensor networks, anonymity and metadata protection mechanisms are not relevant and work in contrast to node authenticity. While other BP implementations, such as IBR-DTN, need to take extra care to secure the lower network layer, e.g., TCP or UDP, µDTN works directly above the data link layer. Thus, only Denial-of-Service jamming attacks against the physical layer must be considered additionally to µDTNSec.

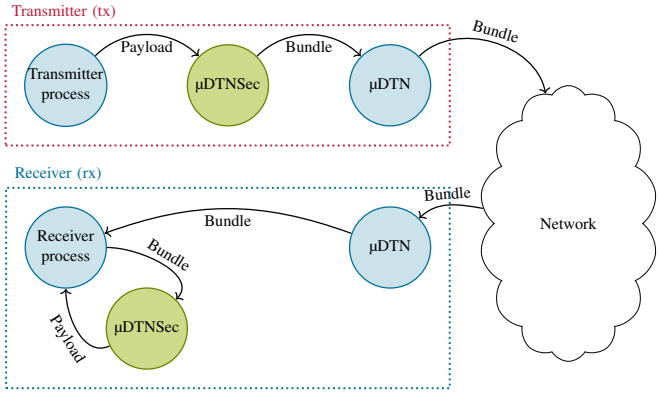


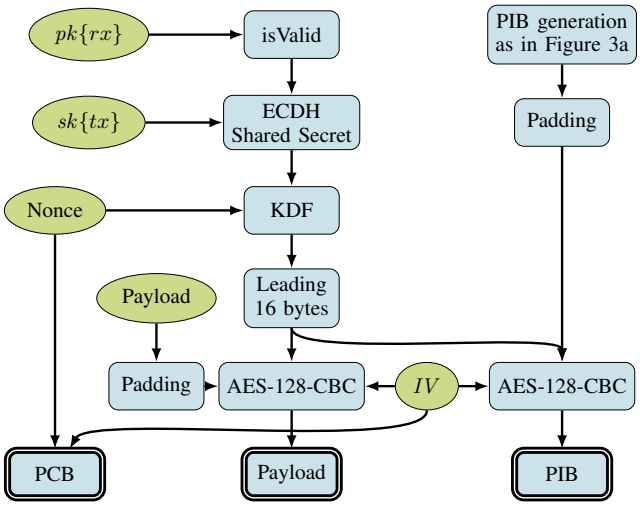
Fig. 1: Interaction between µDTN and µDTNSec

Finally, attacks against the hardware, which require physical access to a placed node, are considered out of scope. This includes tampering with the hardware and side channel and fault attacks, such as key extraction by power measurements.

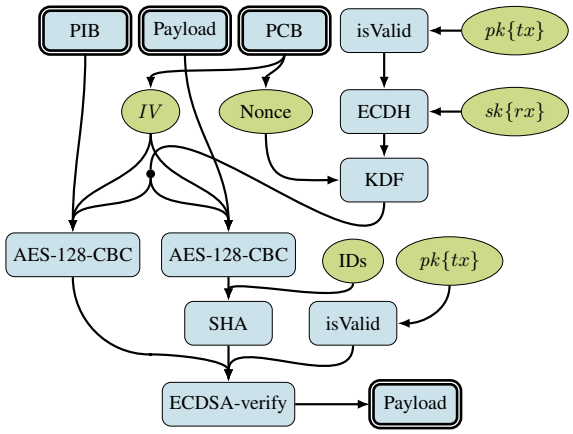
C. System Design

µDTNSec has been developed for the INGA sensor node platform [31]. INGA is equipped with a AT86RF23X radio chip that already provides a hardware implementation of AES-128 in Electronic Code Book (ECB) and CBC mode. Interfaces to its AES methods and RNG have already been implemented for Contiki during the design of RAIM [22]. ECC is implemented in software using the NanoECC [15] and AVR-Crypto-Lib [29] libraries.

Previously, in the µDTN program flow, Contiki processes were able to call a µDTN function to transmit a payload together with the destination (receiver) IDs. µDTN encapsulated the given data into a bundle according to RFC 5050 for Store-Carry-Forward routing. We integrate µDTNSec in this flow as depicted in Figure 1. On the transmitting side, Contiki processes should now call µDTNSec's function instead of µDTN. This allows µDTNSec to apply the configured security protection before handing the encapsulated bundle to µDTN. On nodes forwarding a bundle on the route to the final destination (receiver), bundles can still be processed by µDTN. This is possible because µDTNSec provides end-to-end security only, i.e., security functions only need to be called by the source and the final destination. This also means that no additional processing overhead is required by forwarding



(a) Transmitter: Encryption and generation of digital signature



(b) Receiver: Decryption and verification of digital signature

Fig. 2: Flow charts of µDTNSec's Sign-then-Encrypt mode: AES encryption/decryption is done in-place inside the payload block/PIB. The PCB contains a nonce and the IV. Digital signatures are encapsulated in the encrypted PIB

nodes. At the final destination (receiver), µDTN processes the bundle and hands it to the Contiki receiver process. If the bundle includes security protection, the process calls µDTNSec for decryption/verification.

D. Key Management

µDTNSec is based on ECC as specified by the Standards for Efficient Cryptography Group (SECG) [32], [30]. It includes the algorithms Elliptic Curve Digital Signature Algorithm (ECDSA) for signatures and Elliptic curve Diffie–Hellman (ECDH) for encryption. For each node, a secret/public key pair sk, pk is generated. Our implementation supports the secp128r1, secp192r1, secp256r1 curves for different security levels [30]. We assume that public keys have been pre-deployed in a secure way beforehand.

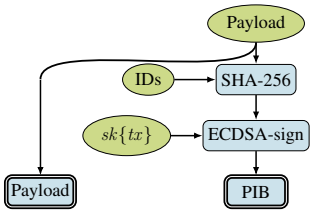
E. Signature Mode

µDTNSec's signature mode provides end-to-end authenticity/non-reputability, which automatically includes integrity protection. We implemented signature generation and verification based on ECDSA with SHA-256. Following the flow chart in Figure 3a, first a SHA-256 hash is calculated over the payload and the non-mutable fields of the bundle. Non-mutable fields are all fields that are not changed by nodes forwarding the bundle on the way to its final receiver. In µDTNSec, the non-mutable fields include the IDs of the transmitter, the transmitter process, the receiver, and the receiver process. This is designed similar to the *Mutable Canonicalization* in the Bundle Security Protocol RFC 6257 [4]. No timestamps are included by µDTN.

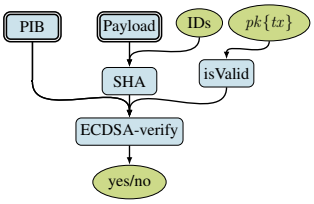
$$h = SHA\text{–}256(payload \parallel non\text{–}mutable\text{–}fields)$$

The resulting hash h is finally signed with ECDSA [33] using the transmitters secret key $sk\{tx\}$ to produce a signature.

$$\sigma = ECDSA\text{–}sign(sk\{tx\}, h)$$



(a) Transmitter: Generation



(b) Receiver: Verification

Fig. 3: Flow charts of µDTNSec's signature mode: Generation and verification of digital signatures encapsulated in the PIB

The signature σ is encapsulated as a Payload Integrity Block (PIB) and handed over to µDTN together with the payload.

The verification of bundles including a PIB works analogously. Following Figure 3b, a hash is calculated over the payload and non-mutable fields. To protect against related key attacks, the public key of the transmitter $pk\{tx\}$ is validated first, i.e., it is checked if the key is based on the correct curve. For verifying σ, ECDSA verification is executed.

$$ECDSA\text{–}verify(pk\{tx\}, h, \sigma)$$

F. Sign-then-Encrypt Mode

µDTNSec's Sign-then-Encrypt mode provides end-to-end confidentiality in addition to the security features provided by its signature mode. Besides ECDSA signatures, this mode uses hardware-backed AES encryption, while its session key is asymmetrically secured with ECDH. Following the flow chart in Figure 2a, a shared secret is derived using ECDH. This requires the receiver's public key $pk\{rx\}$, which has been validated to be based on the correct curve, and the transmitter's secret key $sk\{tx\}$. Because this shared secret would be the same for each communication, an additional

nonce, generated from a secure Random Number Generator, is taken into account. A unique session key is derived using the ANSI-X9.63-KDF [32].

$$s = KDF(ECDH(pk\{rx\}, sk\{tx\}), nonce))$$

A PIB is generated as described in µDTNSec's signature mode. The payload and PIB are symmetrically encrypted with the session key s using AES in CBC mode. It is important to note that symmetric encryption is done in-place without adding new blocks. Here, the low-level operations of AES are provided by the AT86RF23X chip [21]. The utilized CBC mode requires a public initialization vector IV, which should be unique for each bundle. This IV is stored together with the nonce inside the added Payload Confidentiality Block (PCB). The resulting bundle is handed back to µDTN.

After µDTNSec extracts the three blocks, the decryption and verification of bundles works as follows. Following Figure 2b, $pk\{tx\}$ is validated and used together with $sk\{rx\}$ to derive a shared secret. Using the public nonce, the same session key s is derived. Together with the IV, it is used to decrypt the payload and PIB. The signature encapsulated in PIB is verified as described before. Finally, the payload is only returned to the calling process if the verification succeeded.

IV. Evaluation

Our evaluation focuses on the energy and time overhead introduced by the security layer to a traditional µDTN communication. In our evaluation setup we connected two INGA sensor nodes to a precise energy measurement device named Potatoscope [34]. By utilizing this device we were able to trigger the measurement exactly at the time µDTNSec started processing a bundle and also to stop the measurement when µDTNSec finished.

In the first part of the evaluation, we performed a measurement just for the PIB with the secp192r1 curve, in the second part we compare the PIB+PCB processing with three curves namely: secp128r2, secp192r1, and secp256r1. By comparing the two parts we can point out the overhead introduced by the encryption or decryption. The ciphersuites used are the ones mentioned in Table I. To investigate the influence the payload size, we evaluated six different sizes from 1 byte to 250 bytes.

Figure 4 shows the mean time needed to verify the authenticity of a bundle (blue circles). As the values are neither linear rising nor falling and have a standard deviation higher than the span of the of the mean values, we can assume the influence of the payload size for the verification is negligible. The red triangles represent the time needed to process the PIB at the transmitter of the bundle. With up to 7.95 seconds the transmitter needs significantly less time than the receiver with up to 9.51 seconds. These results match with the ones by Jansma et al. [35]. Similar to the receiver, the payload does not have a significant influence.

The comparison between the time in Figure 4 and the consumed energy plotted in Figure 5 shows that both correlate almost perfectly. At the transmitter the energy ranges from $0.1258\,mWh$ to $0.1265\,mWh$. As for the time, the receiver's

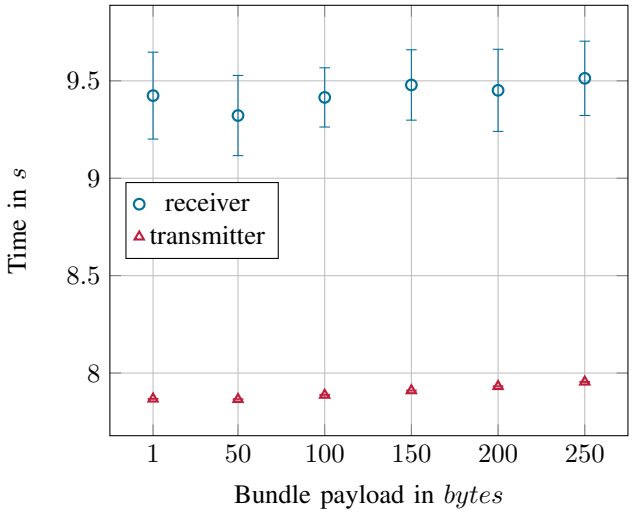


Fig. 4: Signature mode: The red triangles show the time need to generate a PIB with secp192r1 at the transmitter, the blue circles indicate the time used to verify its authenticity at the receiver

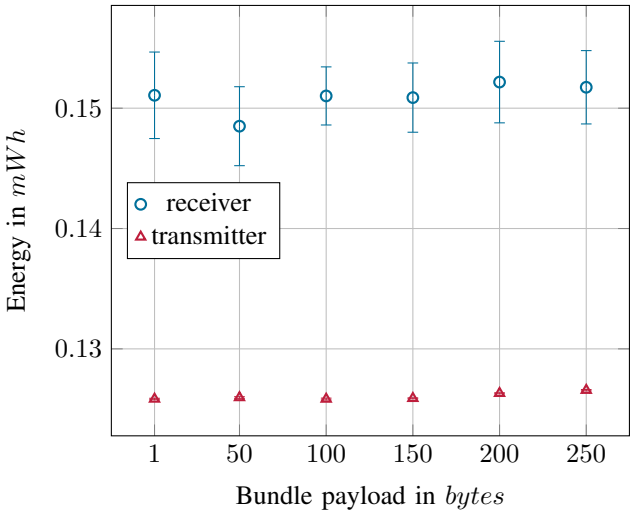
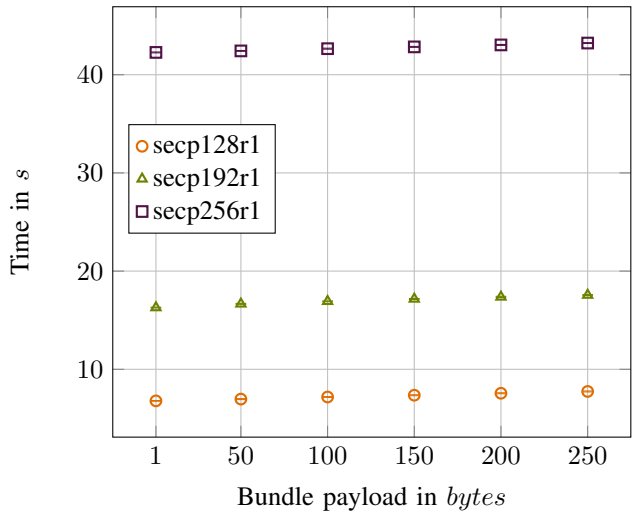


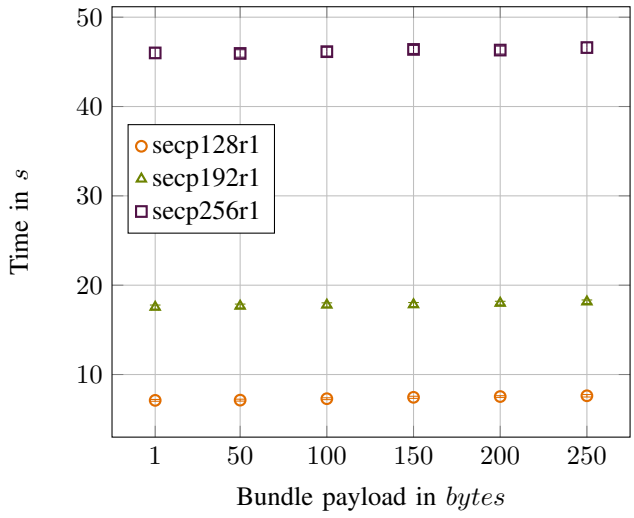
Fig. 5: Signature mode: The red triangles show the energy need to generate a PIB with secp192r1 at the transmitter, the blue circles indicate the energy used to verify its authenticity at the receiver

effort in terms of energy is higher with $0.1484\,mWh$ to $0.1521\,mWh$. Thus, the receiver needs approximately 20 % more energy than the transmitter to process the PIB.

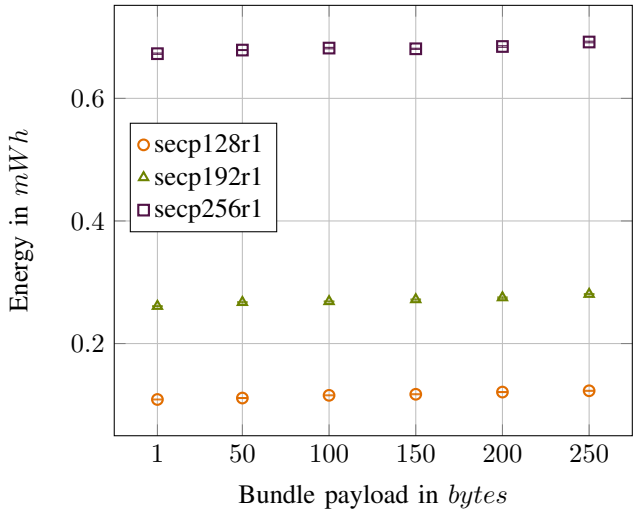
In comparison to the PIB, the processing of PIB+PCB doubles the amount of time and energy needed, as the green triangles in Figures 6a to 6d show. This is due to the additional encryption operation of the payload. Another observation from the figures is that the influence of the chosen curve clearly dominates the influence of the payload size. While the payload's influence is hardly noticeable, secp256r1 doubles the energy and time needed to process a bundle compared to



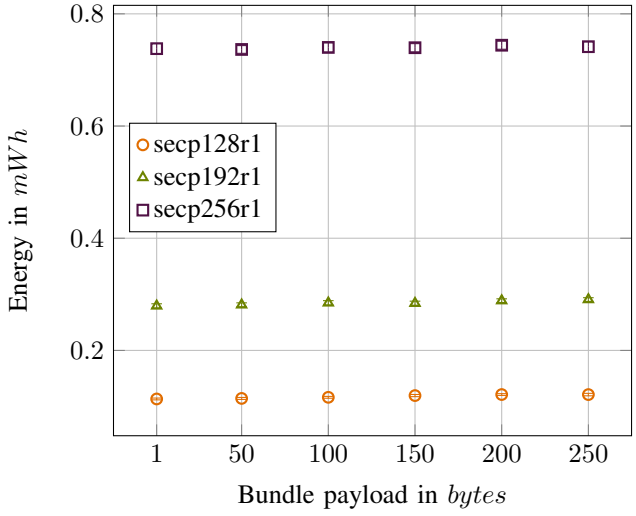
(a) Sign-then-Encrypt mode: Time needed to encrypt and sign a bundle's content at the transmitter for the three evaluated curves



(b) Sign-then-Encrypt mode: Time need to decrypt and verify a bundle's content at the receiver for the three evaluated curves



(c) Sign-then-Encrypt mode: Energy consumed to encrypt and sign a bundle's content at the transmitter for the three evaluated curves



(d) Sign-then-Encrypt mode: Energy consumed to decrypt and verify a bundle's content at the receiver for the three evaluated curves

Fig. 6: Comparison of the influence of payload size and ECC curve choice on energy consumption and performance

secp192r1 and this doubles it from secp128r1. For secp128r1 the receiver needs up to 0.121 mWh to decrypt and verify the payload of Bundle. It lasts 7.621 s to perform this operations. For secp256r1 0.741 mWh and 46.6 s are needed. The values for the transmitter of a bundle are similar: 0.122 mWh and 7.742 s for secp128r2. Utilizing secp256r1 at the transmitter the consumption is raised to 0.691 mWh and 43.227 s.

The results show that the curve should be chosen carefully to the needs of the application. It needs to be considered whether the additional security that secp256r1 offers compensates the five times higher energy and time consumption compared to secp128r1. Further it need to be considered whether an application needs encryption. If data authenticity is sufficient only the PIB should be used, this reduces the energy and time consumption by half.

Another suggestion we deduce from the results is to gather measurement data as long as possible and therefore transmit bundles as big as possible. How long measurement data can be gathered locally depends on the capabilities of the sensor node and the requirements of the application. In longterm patient monitoring for example, sensor data can be gathered over a long time before forwarding it protected by µDTNSec.

Altogether, the evaluation showed that µDTNSec (µDTNSec) needs be configured for its application to perform best. Public-key cryptography introduces significant costs for microcontrollers. Thus, µDTNSec provides no appropriate solution for real-time communications. However, its configurability makes it suitable for a wide range of delay-tolerant applications.

V. Conclusion

We presented µDTNSec, an extension to the BP implementation µDTN for Contiki OS. We have outlined the design decisions as well as the architecture of µDTNSec. Due to the performance and energy impact of ECC, the required security level must be chosen carefully. For example, if only authenticity/integrity is required, µDTNSec's signature mode should be used without encryption.

µDTNSec is licensed as open-source and can be downloaded at https://www.ibr.cs.tu-bs.de/projects/mudtn/.

References

[1] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, Aug. 2014.

[2] S. B. K. Scott, "Bundle protocol specification," RFC 5050, Dec. 2007. [Online]. Available: https://tools.ietf.org/html/rfc5050

[3] G. von Zengen, F. Büsching, W.-B. Pöttner, and L. Wolf, "An Overview of μDTN: Unifying DTNs and WSNs," in *Proceedings of the 11th GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN)*, Darmstadt, Germany, Sep. 2012.

[4] S. Symington, S. Farrell, H. Weiss, and P. Lovell, "Bundle security protocol specification," RFC 6257, May 2011. [Online]. Available: https://tools.ietf.org/html/rfc6257

[5] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "IBR-DTN: A lightweight, modular and highly portable bundle protocol implementation," *Electronic Communications of the EASST*, vol. 37, pp. 1–11, Jan. 2011.

[6] "DTN2 Reference Implementation." [Online]. Available: https://sourceforge.net/projects/dtn/

[7] S. Burleigh, "Compressed Bundle Header Encoding (CBHE)," RFC 6260, May 2011. [Online]. Available: https://tools.ietf.org/html/rfc6260

[8] W.-B. Pöttner, F. Büsching, G. von Zengen, and L. Wolf, "Data elevators: Applying the bundle protocol in delay tolerant wireless sensor networks," in *The Ninth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Las Vegas, Nevada, USA, Oct. 2012.

[9] S. Rottmann, R. Hartung, J. Käberich, and L. C. Wolf, "Amphisbaena: A Two-Platform DTN node," in *The 13th International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2016)*, Brasilia, Brazil, Oct. 2016.

[10] M. Feldmann and F. Walter, "μPCN — A bundle protocol implementation for microcontrollers," in *2015 International Conference on Wireless Communications Signal Processing (WCSP)*, Oct. 2015.

[11] S. N. Rabin Patra, "DTNLite: A Reliable Data Transfer Architecture for Sensor Networks," *CS294-1: Deeply Embedded Networks (Lecture)*, 2003.

[12] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," in *Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2004, pp. 119–132.

[13] NIST, "Recommendation for Key Management," *Special Publication 800-57 Part 1 Rev. 4*, 2016.

[14] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Computer Communications*, vol. 30, no. 11-12, pp. 2314–2341, 2007.

[15] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: Testing the limits of Elliptic Curve Cryptography in sensor networks," in *Wireless Sensor Networks*, ser. Lecture Notes in Computer Science, R. Verdone, Ed. Springer, 2008, vol. 4913, pp. 305–320.

[16] L. B. Oliveira and R. Dahab, "Pairing-based cryptography for sensor networks," in *5th IEEE International Symposium on Network Computing and Applications, Cambridge*, 2006.

[17] L. B. Oliveira, D. F. Aranha, C. P. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab, "TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," *Computer Communications*, vol. 34, no. 3, pp. 485–493, 2011, special Issue of Computer Communications on Information and Future Communication Security.

[18] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography," https://github.com/relic-toolkit/relic.

[19] M. Sethi, J. Arkko, and A. Keranen, "End-to-end security for sleepy smart object networks," in *IEEE 37th Conference on Local Computer Networks Workshops (LCN Workshops)*, Oct. 2012, pp. 964–972.

[20] R. de Clercq, L. Uhsadel, A. Van Herrewege, and I. Verbauwhede, "Ultra low-power implementation of ECC on the ARM Cortex-M0+," in *Proceedings of the 51st Annual Design Automation Conference (DAC)*. New York, NY, USA: ACM, 2014, pp. 112:1–112:6.

[21] Atmel Corporation, "AT86RF231/ZU/ZF datasheet." [Online]. Available: http://www.atmel.com/images/doc8111.pdf

[22] D. Schürmann, F. Büsching, S. Willenborg, and L. C. Wolf, "RAIM: redundant array of independent motes," in *Conference on Networked Systems (NetSys'17)*, Göttingen, Germany, Mar. 2017.

[23] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*. New York, NY, USA: ACM, 2004, pp. 162–175.

[24] R. Doriguzzi Corin, G. Russello, and E. Salvadori, "TinyKey: A light-weight architecture for Wireless Sensor Networks securing real-world applications," in *Eighth International Conference on Wireless On-Demand Network Systems and Services (WONS)*, Jan. 2011, pp. 68–75.

[25] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: a secure sensor network communication architecture," in *6th International Symposium on Information Processing in Sensor Networks (IPSN)*. IEEE, 2007, pp. 479–488.

[26] A. Liu and P. Ning, "TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks," in *International Conference on Information Processing in Sensor Networks (IPSN'08)*. IEEE, 2008, pp. 245–256.

[27] L. Casado and P. Tsigas, "ContikiSec: A secure network layer for wireless sensor networks under the contiki operating system," *Identity and Privacy in the Internet Age*, pp. 133–147, 2009.

[28] F. Büsching and L. Wolf, "The rebirth of One-Time Pads – secure data transmission from BAN to sink," *IEEE Internet of Things Journal*, vol. 2, no. 1, pp. 63–71, Feb. 2015.

[29] 'Bg', "AVR–Crypto–Lib." [Online]. Available: https://wiki.das-labor.org/w/AVR-Crypto-Lib/en

[30] SECG, "SEC 2: Recommended elliptic curve domain parameters," *Standards for Efficient Cryptography Group, Certicom Corp*, Jan. 2010.

[31] F. Büsching, U. Kulau, and L. Wolf, "Architecture and evaluation of INGA - an inexpensive node for general applications," in *Sensors, 2012 IEEE*. Taipei, Taiwan: IEEE, Oct. 2012, pp. 842–845.

[32] SECG, "SEC 1: Elliptic curve cryptography," *Standards for Efficient Cryptography Group, Certicom Corp*, May 2009.

[33] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[34] R. Hartung, U. Kulau, and L. C. Wolf, "Demo: PotatoScope - scalable and dependable distributed energy measurement for WSNs," in *IEEE SECON 2016 Conference Proceedings*, London, UK, Jun. 2016.

[35] N. Jansma and B. Arrendondo, "Performance comparison of elliptic curve and RSA digital signatures," University of Michigan, College of Engineering, Tech. Rep., Apr. 2004.