Forward Secure Delay-Tolerant Networking*

Signe Rüsch
Dominik Schürmann
Rüdiger Kapitza
Lars Wolf
ruesch@ibr.cs.tu-bs.de
schuermann@ibr.cs.tu-bs.de
kapitza@ibr.cs.tu-bs.de
wolf@ibr.cs.tu-bs.de
TU Braunschweig
Institute of Operating Systems and Computer Networks

ABSTRACT

Delay-Tolerant Networks exhibit highly asynchronous connections often routed over many mobile hops before reaching its intended destination. The Bundle Security Protocol has been standardized providing properties such as authenticity, integrity, and confidentiality of bundles using traditional Public-Key Cryptography. Other protocols based on Identity-Based Cryptography have been proposed to reduce the key distribution overhead. However, in both schemes, secret keys are usually valid for several months. Thus, a secret key extracted from a compromised node allows for decryption of past communications since its creation.

We solve this problem and propose the first forward secure protocol for Delay-Tolerant Networking. For this, we apply the Puncturable Encryption construction designed by Green and Miers, integrate it into the Bundle Security Protocol and adapt its parameters for different highly asynchronous scenarios. Finally, we provide performance measurements and discuss their impact.

CCS CONCEPTS

• **Networks** → **Security protocols**; *Network mobility*; *Mobile networks*; *Peer-to-peer networks*;

KEYWORDS

Forward Secrecy, Puncturable Encryption, Delay-Tolerant Networking, DTN

ACM Reference Format:
Signe Rüsch, Dominik Schürmann, Rüdiger Kapitza, and Lars Wolf. 2017. Forward Secure Delay-Tolerant Networking. In *Proceedings of CHANTS'17*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3124087.3124094

*Updated version (2017-10-15) with corrected values for the vehicular scenario.

CHANTS'17, October 20, 2017, Snowbird, UT, USA

ACM ISBN 978-1-4503-5144-7/17/10…$15.00
https://doi.org/10.1145/3124087.3124094

1 INTRODUCTION

In typical computer networks, end-to-end connectivity can be assumed as a basic feature. However, in certain scenarios, such as InterPlanetary Networks (IPNs), communication systems for rural villages, or delay-tolerant vehicular networks, this assumption does not hold. Nodes can move out of communication range or environmental factors may disturb the channel. To overcome these obstacles, Delay-Tolerant Networks (DTNs) [6] have been designed usually based on the Bundle Protocol specification [21]. Here, data is encapsulated using self-contained bundles that are transmitted using the *Store-Carry-Forward* approach. DTNs have many areas of application and the security of the transmitted bundles is an important feature. The Bundle Security Protocol [23] defines bundle types for end-to-end and hop-to-hop security that offer confidentiality, integrity, and authenticity using a traditional Public-Key Infrastructure (PKI). In recent years, Identity-Based Cryptography (IBC) gained popularity as it eliminates the need for public-key distribution prior to encryption [2, 13].

However, all existing schemes are vulnerable to the following attack: An attacker can passively record all encrypted bundles intended for a specific node and decrypt them at a later point in time when the node's secret key is leaked. This is highly probable, because at some point, software exploits or design flaws will be found in real-world implementations allowing the extraction of secret keys remotely or via physical access. A naïve protection against this attack works by encrypting each bundle with a different ephemeral key. An attacker would have no common key for previously received bundles and could therefore not decrypt them. This security property is called forward secrecy. As depicted in Figure 1, it provides protection of past communication until a specific point in time.

Nowadays, many websites enable forward secure TLS ciphersuites to encrypt each session with a different ephemeral key [8].

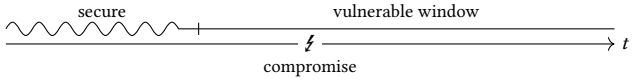


Figure 1: Forward Secrecy provides protection of past communication until the point in time where old ephemeral keys are deleted (Source: [24])

For asynchronous communication, however, it is much harder to include forward secrecy without reducing usability, especially in mobile communication. For that reason, it is often either forward secrecy or asynchronous communication but not both. Neither OpenPGP [4] nor S/MIME [17], two popular end-to-end encryption protocols for email communication, provide forward secrecy.

Similar to email communication, DTNs are used in highly asynchronous scenarios, where no direct connection between sender and recipient exist. In addition, nodes are often highly mobile and connections are not lasting forever. Bundles can be lost or received in a different order as they have been sent. In PKI-based DTNs, public keys are often distributed in the network after their creation to provide them to other nodes for encryption. Flooding the network with new keys is an expensive and error-prone operation. Thus, secret keys are typically valid for up to two years. Even in IBC-based DTNs, keys are valid for several months and can only be renewed after directly connecting to a so called Private Key Generator (PKG) because new key pairs cannot be created autonomously [2]. Conclusively, it is impractical to generate new key pairs for short time periods in DTNs.

In this paper, we present a way to make DTNs forward secure using the Puncturable Encryption construction by Green and Miers [11]. We integrate it into the Bundle Security Protocol and evaluate possible parameter configurations for different scenarios. By implementing the libforwardsec library in IBR-DTN [19], we are able to provide performance measurements. Finally, we discuss its impact in comparison to traditional RSA ciphersuites defined in the Bundle Security Protocol.

Our implementation is fully open source and can be downloaded from https://www.ibr.cs.tu-bs.de/projects/ibr-dtn/.

2 RELATED WORK

One topic that is often discussed regarding security in DTNs is key management where the different approaches using PKI- and IBC-based systems are compared. Overall, it can be seen that IBC-based systems have some advantages compared with traditional PKI-based systems in DTN environments, but the practical success is not as clear as the conceptual design may suggest [2]. IBC-based systems have certain small advantages regarding confidentiality in DTN environments; however, the higher computational cost especially for resource-constraint devices defeats these advantages.

In 2015, Green and Miers [11] presented a method to ensure forward secrecy in which they introduce Puncturable Encryption. Using the puncture algorithm, a new secret key is created and certain previously defined tags are selected so that messages with these tags cannot be decrypted anymore using the new key. The messages are therefore protected from potential attackers. This method leverages multiple other algorithms such as an hierarchical IBC scheme by Boneh et al. [3], a public key encryption scheme with forward secrecy by Canetti et al. [5], and Non-Monotonic Attribute Based Encryption (NM-ABE) by Ostrovsky et al. [16]. The latter is necessary for the puncture algorithm that is used to modify the secret key to ensure forward secrecy. The authors also provide the library libforwardsec that implements this forward secure encryption scheme.

Some approaches to combine asynchronous communication and forward secrecy already exist, for example the Signal protocol [7]. Here, a client A uploads 100 signed key exchange messages, so-called prekeys, to a server, which can then be used by another client B that wants to communicate with A. However, if this would be adapted for DTNs without any centralized server, prekeys need to be distributed in the network requiring memory storage on all participating nodes. Furthermore, the procedure would be prone to DoS attacks where an attacker consumes the entire supply of prekeys preventing others from encrypting messages to the corresponding node [11]. Extending, e. g., email, to include forward secrecy often comes with an increase in complexity [20, 22]. Most proposals require a highly available infrastructure to exchange fresh key material. This is an even bigger issue in DTN environments. Therefore, the procedure of Green and Miers [11] is a great possibility to include both forward secrecy and asynchronous communication in DTNs without added complexity or infrastructure.

Recently, a modified encryption scheme based on this work by Green and Miers presents a solution to forward secrecy for 0-RTT protocols [12]. 0-RTT protocols seek to reduce the number of round trips necessary for the TLS key exchange before encrypted application data can be exchanged. TLS 1.3 [18] employs 0-RTT protocols; however, it was assumed to be impossible to ensure forward secrecy for the first round trip message of the TLS handshake without a previously shared state [14]. The scheme of Günther et al. [12], however, leverages the forward secure Puncturable Encryption scheme to achieve this.

Our focus lies on the application of the forward secure Puncturable Encryption scheme in DTNs and thereby ensuring forward secrecy for all bundles exchanged between nodes. We are not aware of other work providing forward secrecy in DTNs.

3 FORWARD SECURE ENCRYPTION

Forward secure encryption means that the user's secret key is periodically updated to ensure that past messages remain confidential in the event of a compromised key [11]. The general idea behind Puncturable Encryption is that recipients may repeatedly update the decryption keys to revoke decryption capabilities for selected messages, recipients, or time periods without the necessity of a new key exchange. In this scheme, time intervals are mapped to portions of the secret key. This is achieved by combining Puncturable Encryption with a forward secure public key encryption scheme (FS-PKE) by Canetti et al. with the goal of practical forward secure messaging with a low overhead [5].

The encryption model assumes a sender and a receiver who interact via an insecure channel, an existing PKI to exchange keys, and that sent messages are either initial or interactive. The focus here is on asynchronous communication, as forward secrecy for interactive, synchronous communication can easily be ensured by other protocols such as OTR [11].

Puncturable Encryption is a form of tag-based encryption using short, unchanging public keys and includes a puncture algorithm. This *puncturing* can be described as follows: on input of the current secret key SK and a tag $t \in \{0, 1\}$, it outputs a new SK' that will decrypt all ciphertexts that are not encrypted under that tag t. It is based on the Attribute-Based Encryption (ABE) scheme by

Ostrovsky et al. [16]. This is combined with the FS-PKE scheme [5], which allows for revocation of decryption capabilities for certain intervals without revocation of all previous intervals. FS-PKE uses Hierarchical Identity Based Encryption (HIBE) as building blocks, employed here is the optimized scheme by Boneh et al. [3]. The total decryption and key storage cost grows linearly only in the maximum number of messages received within a given time period and logarithmically in the number of time periods.

An active attacker may block messages so that the recipient cannot revoke decryption capabilities; however, using the combination of the two schemes that allow both revocation for tags and time intervals, it is possible to ensure that the message cannot be decrypted after a certain decryption window has passed. This requires that the secret keys of both schemes are cryptographically bound to each other.

The algorithms of this combined scheme, called the Forward Secure Encryption (FSE) scheme, are as follows [11]:

- $KeyGen(1^d, k) \rightarrow (PK, SK_0)$: given a security parameter d and a maximum number of tags k, it outputs a new public key and an initial secret key
- $Encrypt(PK, M, t_1, \ldots, t_k) \rightarrow$ ciphertext CT
- $Decrypt(PK, SK_i, CT, t_1, \ldots, t_k) \rightarrow \{M\} \cup \{\perp\}$
- $Puncture(PK, SK_{i-1}, t) \rightarrow SK_i$: see Section 3.1
- $NextInterval(SK_n)$: see Section 3.2

During the key pair generation, first the parameters for an instance of the ABE scheme are produced, the public parameters are published as the public key, and a decryption key is derived from the master secret key which is then destroyed. The user now has a pair of two secret keys (A_T, B_T), where A_T is the key for the FS-PKE scheme and B_T the one for the puncturable encryption scheme, for each time interval T. For puncturing on a tag t, a new secret key containing the negation of t is derived. Messages are encrypted for certain time intervals and with one or multiple tags; they can then only be decrypted with the secret key that corresponds to the same time interval and that is not punctured on any of the attached tags.

Both schemes are provably secure under the Decisional Bilinear Diffie-Hellman Inversion and the Decisional Bilinear Diffie-Hellman assumption for bilinear groups.

3.1 Puncturing of Keys

The general puncturing algorithm is shown in Figure 2. To each (initial) ciphertext, a unique identifier, the 'tag', is attached, which is generated by the sender. When a user receives such a tagged ciphertext, they may revoke their secret key's decryption capabilities for this tag via a secret key update, i.e. by *puncturing* on that tag. It is also possible to use multiple tags per ciphertext, e. g., for GUIDs, counters, tags about subjects, or sender IDs. This way, one may, for example, puncture on all ciphertexts from one sender, all regarding one subject, on all in general, or only a single ciphertext, thereby disabling future decryption of these ciphertexts.

3.2 Key Forwarding

The lifetime of the FSE key is divided into several time intervals of steady, but variable length. *Key forwarding* describes the process of deriving new secret keys for a new time interval. Old keys can either be discarded or kept in the case of late arrivals, they can also be sterilized so that they can still decrypt but cannot be used for deriving any more new keys. Tags can only be punctured on if keys for the next interval have already been derived, so until then messages are not secure according to forward secrecy. It is also not advisable to store keys for past intervals indefinitely, as these interval keys may be compromised.

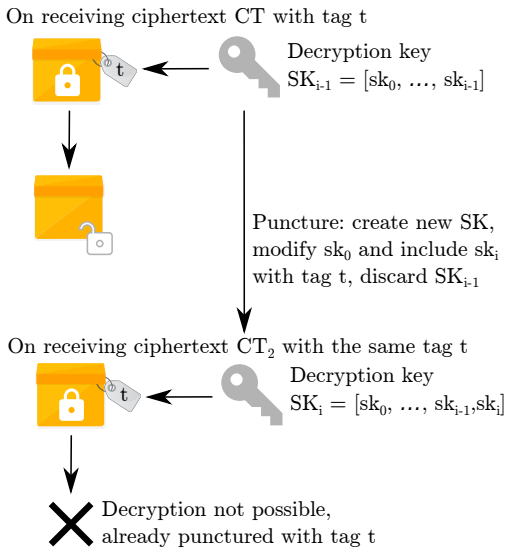


Figure 2: After receiving a ciphertext with tag t, the secret key can be punctured: a new secret key is created which cannot decrypt messages with the same tag anymore.

4 FORWARD SECURE DTNS

After introducing the FSE construction, we apply it to DTNs. We discuss the integration of this new scheme into the Bundle Security Protocol and discuss a way to generate unique tags compatible with the existing Bundle Protocol specification. Because FSE needs to be configured based on the targeted DTN scenario, we give an overview over all important parameters. Finally, we provide implementation specific considerations.

4.1 Extending the Bundle Protocol

We seek to extend the security guarantees offered by the Bundle Security Protocol by forward secrecy for every bundle that is transmitted between sender and receiver. Using the FSE scheme presented in Section 3, bundles are equipped with a unique tag and on reception of a bundle, the recipient's secret key is punctured on attached tag, thereby revoking decryption capabilities. To support this scheme, it is added as an alternative to the common ciphersuites of the Bundle Security Protocol, without any further modification of the defined bundle types. The key management remains unchanged, only now instead of RSA keys, the FSE public keys are exchanged. However, the key management itself is out of scope and we assume some form of key exchange prior to communication. Signature creation and verification is also not considered here as they are not included in the FSE scheme.

4.2 Unique Tags in DTNs

In the FSE scheme, each message can be equipped with one or multiple tags, which can, for example, be used to uniquely identify messages or group them according to subject. For generic DTN communication, we assume that every bundle should be uniquely identifiable by its tag and decrypted once by the receiver. Thus, we propose to generate the tag based on the hash of node's EID and the current timestamp, e. g., for node A, $t = h$("A-2017-06-04") and $k = 1$ (cf. Section 3). Immediately afterwards, decryption of this bundle should be permanently disabled by puncturing the FSE key on this tag. Each bundle can be assumed to be unique within their source endpoint, creation timestamp, and the creation timestamp sequence number [21]. We therefore calculate the hash of these values to be used as unique tags for bundles, to offer forward secrecy separately to each bundle.

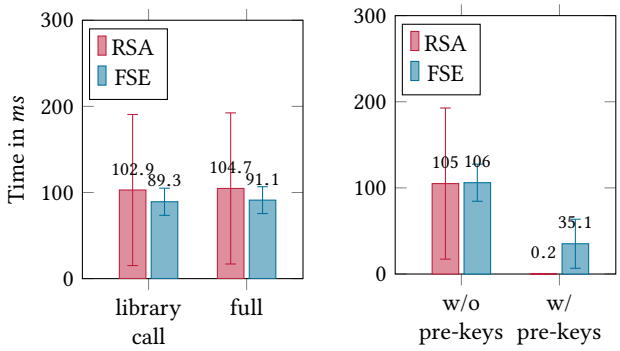
4.3 FSE Parameters

The FSE scheme in general is based on four parameters introduced in Section 3 and Section 4: The **interval length** n describes the lifetime of an FSE key which is divided into time intervals of steady, but variable length. For optimal performance of encryption and decryption, it is recommended to receive one message per interval, i.e., perform one puncturing operation and immediately forward the key [11]. Messages can be attached with a certain **amount of tags** $k \geq 1$, the default behaviour of libforwardsec is to include one tag per message. FSE keys support a certain **amount of time intervals** d, which means that after d time intervals, new FSE keys have to be generated and exchanged. A maximum of 2^{31} time intervals is supported by libforwardsec. As keys are periodically forwarded and thereby lose the ability to decrypt messages of prior intervals, old interval keys should be retained to support late or early arrival of messages. N defines how many **currently valid interval keys** are stored, i.e. how many secret keys of past intervals in addition to the current one are preserved. This is advantageous for lost bundles, which are sent but never received. They will be encrypted for a specific time interval; however, as secret keys will not be stored indefinitely, the corresponding secret key will be deleted after N time intervals have passed, thereby guaranteeing forward secrecy even for these bundles. For special scenarios, keys could even be derived in advance before usage.

The optimal choice of these values for different network types is heavily influenced by both their environment and the chosen applications. We provide an evaluation for example scenarios in Section 5.2.

4.4 Implementation

The encryption scheme offering forward secrecy is implemented in the library libforwardsec, which is openly available [15]. It is written in C++ and uses both the Relic pairing library and the Cereal serialization library, which wraps Relic's serialization routines for elliptic curve points, and for performance optimization it employs OpenMP. The setting employed by libforwardsec uses 256-bit Barreto-Naehrig curves and, in the conversion from symmetric to asymmetric curves, optimizes for minimal ciphertext size. The assumption behind this is that secret key storage is unproblematic except on highly constraint devices, which is acceptable in most



(a) Key Generation (10 repetitions) **(b) Start-up time of `SecurityKeyManager` (10 repetitions)**

Figure 3: Performance of cryptographic operations

DTN environments. We included the FSE scheme in the DTN implementation IBR-DTN. The key management required most changes: the `SecurityKeyManager` of IBR-DTN was extended to support the FSE scheme without modification of the bundle structure specified in the Bundle Security Protocol. The FSE parameters are set during start-up of the IBR-DTN daemon, which implements the DTN node functionality, and sender and receiver are assumed to be on the same time interval. Both the FSE keys and the time interval are restored if the daemon has been started previously; otherwise, new FSE keys are generated. The bundle itself is encrypted with a symmetric session key using AES-256, this session key is then encrypted with the FSE key and attached to the message. After decryption, the daemon punctures the FSE key on the extracted tag for this encryption interval.

5 EVALUATION

We present a microbenchmark demonstrating the performance of IBR-DTN using libforwardsec and an analysis of the optimal FSE parameters for DTN environments. For this, we consider three scenarios: IPNs, communication scenarios for rural villages, and delay-tolerant vehicular networks.

5.1 Microbenchmark

The measurements were conducted on a Dell OptiPlex 7010 Desktop-PC with an Intel Core i7-3770 CPU @ 4(8)x3.4 GHz CPU and 16 GB of RAM running Ubuntu 14.04 LTS. All main key operations were inspected for both the RSA and the FSE scheme, which includes key generation and the start-up time of the `SecurityKeyManager` as well as en- and decryption of bundles.

Key Generation. In Figure 3a, the duration of the key generation for the FSE and RSA schemes is shown. We consider both the complete key generation including serialization of the keys as well as only the duration of the library call to OpenSSL. Here, it can be seen that even though OpenSSL is trimmed for performance, the FSE key generation is faster than that of RSA keys with 89 *ms* to 103 *ms*, respectively, thereby improving performance by 13%. This and RSA's large standard deviation in Figure 3 are due to the fact that for RSA key generation, large prime numbers have to be found.

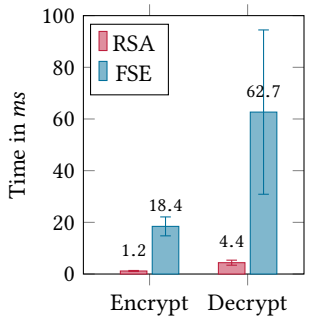
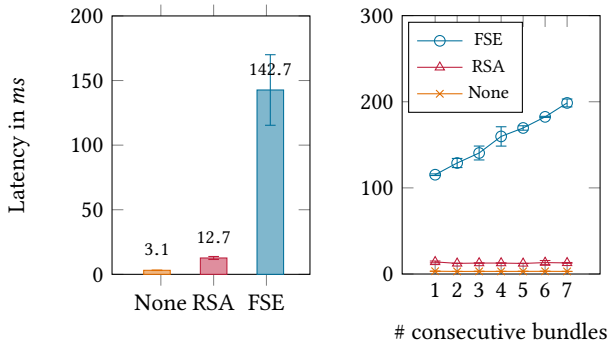


Figure 4: Encryption / Decryption (20 repetitions)



(a) Latency introduced by FSE (30 repetitions)

(b) Latency during interval progression (3 repetitions)

Figure 5: Performance of cryptographic operations

Startup Time of `SecurityKeyManager`. Figure 3b shows the start-up time of the `SecurityKeyManager`. Here, it is differentiated between start-up when the node already possesses a pre-existing key pair which has to be deserialized, and without any pre-existing keys, which in this case have to be generated first and then serialized. The performance difference between deserializing pre-existing RSA keys and FSE keys is negligible.

Encryption and Decryption. The performance of the cryptographic operations for encryption and decryption are shown in Figure 4. Here, the FSE scheme is less performant than the RSA scheme: the duration is about 15.7 times and 14.27 times higher for encryption and decryption, respectively. The FSE decrypt call also contains a call to the puncturing with the received tag, which is done after every message.

Latency. The FSE scheme also introduces a higher latency, as can be seen in Figure 5a. For this, we utilized the DTN application `dtnping` to send ping messages between the nodes. Here, the same trend is apparent as in the other measurements: the RSA scheme is on average about 11 times faster (13 *ms* to 143 *ms*) than the FSE scheme, and without any security features, the operation is about 48 times faster (3 *ms* to 143 *ms*). The performance of cryptographic operations in libforwardsec decreases during the progression of an interval due to the increasing key size [11], which explains the large standard deviation in Figure 4. This leads to an increase in latency over the course of an interval, where round trip times of up to 200 *ms* are possible. Figure 5b shows this increase for the FSE scheme, as every bundle leads to an additional puncture operation. As soon as a new interval is started, the RTT decreases to 115 *ms*.

Summary. These results clearly show the expected better performance of OpenSSL's RSA scheme implementation in comparison to libforwardsec's non-optimized FSE construction. One surprising result was the better performance of libforwardsec regarding key generation. The increasing duration of cryptographic operations during the progression of an interval, which is mentioned by Green and Miers [11], is perceptible in the latency of applications such as `dtnping`.

5.2 Choice of FSE Parameters

In the following we discuss three potential DTN scenarios, namely IPNs, communication systems for rural villages, and delay-tolerant vehicular networks. For all scenarios we propose the usage of $k = 1$, i.e., defining a unique single tag per bundle as specified in Section 4.2. Based on the variance of the delay, the number of currently valid intervals N must be chosen carefully to still allow decryption of bundles arriving late. Thus, we propose to set $N = \lceil Max/Mean \rceil + 1$. Using statistics from existing DTN evaluations, we deduce FSE parameters for each of them. A summary of these results can be found in Table 1.

IPN. First, we consider an IPN scenario by Apollonio et al. [1]. A moon lander sends bundles to a user on earth via the following hops: The moon lander has periodic connections to a satellite orbiting the moon that in turn has periodic connections to a Mission Control Center (MCC) or an Auxiliary Terrestrial Gateway. The earth stations are connected and the end-user communicates through the Internet with the MCC. The contact plan of the involved nodes is fully known. We discuss the streaming scenario where 5 kB bundles are send periodically every 10 s. For this, we propose to set the interval length $n = 124$ s, which is the mean delay between the sender and receiver (cf. Table 1). The secret key is punctured on each decrypted bundle ($k = 1$) to obtain the highest forward secrecy. With this configuration, all bundles are either received during the same interval they have been sent or at least two intervals later, because even if they are sent at the end of the current interval, the worst case delay is less than twice the interval duration n. Thus, the number of stored secret keys, i.e., the currently valid intervals, is set to $N = 3$ following the proposed equation. Conclusively, following the results by Apollonio et al. [1], 5 bundles are received per interval. In the worst case a burst can lead to 11 bundles per interval [1]. As this corresponds to the number of punctures done by the receiving side, the duration of cryptographic operations is still acceptable as shown in Figure 5b.

Rural Village. Grasic and Lindgren [10] deploy communication services to the rural remote village Staloluokta in Sweden. Services in Staloluokta are provided via a data mule helicopter that commutes daily from Ritsem, a camp by the Sami people that is connected to the Internet. Two outdoor nodes are connected via a wifi bridge, where one is located near the helicopter landing place for receiving data from the mule. They provide direct access to

Table 1: Selected DTN scenarios and their corresponding choice of FSE parameters

	Scenario				FSE Parameters		
	Ref.	Description	Mean	Max	n	N	Bundles/Interval
IPN	[1], Fig. 6	Satellite stream from moon to earth	124 s	153 s	124 s	3	5
Rural Village	[10], Tab. 1	Communication for rural villages	87 071 s ≈ 1 d	-	1 d	3	~ 9
Vehicular (a)	[9], Fig. 7	Routing in urban public transport	749 s ≈ 13 min	5900 s ≈ 98 min	13 min	9	~ 1560
Vehicular (b)	—— " ——	—————— " ——————	——— " ———	——— " ———	1 min	99	~ 120

communication services, such as DTN Facebook and messaging to up to 13 end-user nodes (laptops, mobile devices) with an average bundle size of 51 kB. Thus, the scenario is quite static with a mean delay of 1 day due to the commuting helicopter. According to Grasic and Lindgren, 6107 bundles have been sent during 53 days. Thus, on average the ratio was 115 bundles per day and 9 bundles per day per device considering a uniform usage. Conclusively, for this number of punctures the duration of cryptographic operations is still acceptable.

Delay-Tolerant Vehicular Networks. Doering et al. [9] simulate a scenario for transportation systems, e.g., busses, with known timetables and network maps. In their fixed scenario, they considered a routing graph with 54 bus stops, 13 lines, and 28 vehicles. Their routing algorithm called RUTS outperformed other algorithms due to these known parameters. Here, we consider RUTS as an example for a fixed network with high traffic and high variance due to variable hop counts until a bundle reaches a destination. For the simulation, random nodes are sending to other random nodes in a constant interval. Due to a mean delay of 13 min and a maximum delay of 98 min, we choose $N = 9$. Considering a high network traffic to exchange current vehicle positions and traffic jams with ~ 2 bundles per second, the *Vehicular (a)* scenario processes ~ 1560 bundles per interval. This would increase the decryption time significantly to up to ~ 21.6 s and is thus impractical for real world deployments. In comparison to the previous scenarios, this exhibits much higher traffic patterns and high maximum delay times. As a trade-off between low decryption times and degree of forward secrecy, we propose a second configuration *Vehicular (b)* with $n = 1$ min and $N = 99$. In the worst case, this has only ~ 1.8 s decryption time.

6 CONCLUSION

We combined IBR-DTN and libforwardsec to add forward secrecy as a security guarantee to a DTN implementation. Bundles can now be encrypted using the FSE scheme by Green and Miers [11], which, combined with the added interval management and the use of the puncture algorithm, ensures forward secrecy of messages. Our evaluation shows that, while the cryptographic operations introduce an acceptable performance overhead, the latency is considerably higher and increases over the course of a time interval. We show how this can be remedied by choosing suitable values for the parameter of the FSE scheme. However, it has to be weighted based on specific scenario requirements whether forward secrecy outweighs higher latencies.

REFERENCES

[1] P. Apollonio, C. Caini, and V. Fiore. 2013. From the Far Side of the Moon: Delay/Disruption-Tolerant Networking Communications via Lunar. *China Communications* 10, 10 (Oct. 2013), 12–25.

[2] N. Asokan, Kari Jostiainen, Philip Ginzboorg, Jörg Ott, and Cheng Luo. 2007. Applicability of Identity-Based Cryptography for Disruption-Tolerant Networking. In *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking*. 52–56.

[3] D. Boneh, X. Boyen, and E.-J. Goh. 2005. *Hierarchical Identity Based Encryption with Constant Size Ciphertext.* Springer, Berlin, Heidelberg, 440–456.

[4] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. 2007. OpenPGP Message Format. RFC 4880 (Proposed). (Nov. 2007). http://www.ietf.org/rfc/rfc4880.txt

[5] R. Canetti, S. Halevi, and J. Katz. 2003. *A Forward-Secure Public-Key Encryption Scheme.* Springer, Berlin, Heidelberg, 255–271.

[6] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. 2007. Delay-Tolerant Networking Architecture. RFC 4838 (Informational). (April 2007). http://www.ietf.org/rfc/rfc4838.txt

[7] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. 2017. A Formal Security Analysis of the Signal Messaging Protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. 451–466.

[8] T. Dierks and E. Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed). (Aug. 2008). http://www.ietf.org/rfc/rfc5246.txt

[9] M. Doering, T. Pögel, and L. Wolf. 2010. DTN Routing in Urban Public Transport Systems. In *Proceedings of the 5th ACM Workshop on Challenged Networks (CHANTS '10)*. ACM, New York, NY, USA, 55–62.

[10] S. Grasic and A. Lindgren. 2014. Revisiting a Remote Village Scenario and its DTN Routing Objective. *Computer Communications* 48 (2014), 133–140.

[11] M. D. Green and I. Miers. 2015. Forward Secure Asynchronous Messaging from Puncturable Encryption. In *2015 IEEE Symposium on Security and Privacy*. 305–320.

[12] F. Günther, B. Hale, T. Jager, and S. Lauer. 2017. *0-RTT Key Exchange with Full Forward Secrecy.* Springer, Cham, 519–548.

[13] A. Hennessy and A. Alford. 2013. Demo: Identity-Based Cryptography in Delay-Tolerant Networks. In *ExtremeCom '13*.

[14] H. Krawczyk. 2005. *HMQV: A High-Performance Secure Diffie-Hellman Protocol.* Springer, Berlin, Heidelberg, 546–566.

[15] I. Miers. 2015. Libforwardsec. Forward Secure Encryption for Asynchronous Messaging. (2015). https://github.com/imichaelmiers/libforwardsec

[16] R. Ostrovsky, A. Sahai, and B. Waters. 2007. Attribute-Based Encryption with Non-Monotonic Access Structures. In *ACM CCS '07*. 195–203.

[17] B. Ramsdell and S. Turner. 2010. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751 (Proposed). (Jan. 2010). http://www.ietf.org/rfc/rfc5751.txt

[18] E. Rescorla. 2016. The Transport Layer Security (TLS) Protocol Version 1.3. https://tools.ietf.org/html/draft-ietf-tls-tls13-18. (March 2016).

[19] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf. 2011. IBR-DTN: A Lightweight, Modular and Highly Portable Bundle Protocol Implementation. *Electronic Communications of the EASST* 37 (Jan. 2011), 1–11.

[20] B. Schneier and C. Hall. 1997. An Improved E-mail Security Protocol. In *13th Annual Computer Security Applications Conference*. 232–238.

[21] K. Scott and S. Burleigh. 2007. Bundle Protocol Specification. RFC 5050 (Experimental). (Nov. 2007). http://www.ietf.org/rfc/rfc5050.txt

[22] H.-M. Sun, B.-T. Hsieh, and H.-J. Hwang. 2005. Secure E-mail Protocols Providing Perfect Forward Secrecy. *IEEE Communications Letters* 9, 1 (Jan. 2005), 58–60.

[23] S. Symington, S. Farrell, H. Weiss, and P. Lovell. 2011. Bundle Security Protocol Specification. RFC 6257 (Experimental). (May 2011). http://www.ietf.org/rfc/rfc6257.txt

[24] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. 2015. SoK: Secure Messaging. In *IEEE Symposium on Security and Privacy*. 232–249.