APBE: Anti-Piracy Security for SPARTN Augmented GNSS Services

Pericle Perazzo
Department of Information Engineering
University of Pisa
pericle.perazzo@unipi.it

Carlo Vallati
Department of Information Engineering
University of Pisa
carlo.vallati@unipi.it

Davide Lenzarini
U-Blox
davide.lenzarini@u-blox.com

Abstract

Global Navigation Satellite Systems (GNSS) are a key enabler for many new technologies, ranging from autonomous vehicles to shared mobile devices. In order to ensure high precision for those applications, GNSS augmentation systems are needed to provide correction data to reach a precision that is in the order of centimeters. Those systems can be provided as paid services where correction data are broadcast over a satellite link. In order to protect those systems and restrict their access only to paid users, encryption mechanisms must be adopted. SPARTN is an open industry standard for GNSS augmentation that has been specifically designed for supporting encryption, while saving bandwidth on the satellite link. In this paper, we propose APBE (Anti-Piracy Broadcast Encryption), a method to enhance the SPARTN security by providing protection against pirate customers that is specifically tailored to minimize bandwidth and storage. The proposed approach is demonstrated to be feasible via a real proof-of-concept implementation based on an embedded system. APBE is a candidate mechanism to be included in the future versions of SPARTN.

1 Introduction

Global Navigation Satellite Systems (GNSS) are now part of a wide range of applications to provide accurate, continuous and global position, velocity and time information. The system is used in a wide range of applications ranging from navigation to position tracking [10]. Recently, GNSS augmentation technologies have been introduced, in order to provide centimeter-level accurate position information to end users. This technology is a key enabler for a wide range of use cases and applications. A typical example is autonomous mobility: unmanned autonomous vehicles require instantaneous high-precision positioning to avoid accidents or respect geo-location boundaries [4]. Another example is e-micromobility where shared mobile vehicles, e.g., e-scooters, are required to comply with local laws and regulations with high precision for their regular operations and parking[1].

The Secure Position Augmentation for Real Time Navigation (SPARTN) is an open industry standard for GNSS augmentation[2][11, 6]. The standard is specifically designed for supporting encryption to protect the augmentation data. It aims at providing high reliability and accuracy, along with ensuring low-bandwidth in order to work also on satellite channels. SPARTN-based GNSS augmentation is usually provided as a paid service by different companies. One example is the PointPerfect®[3] service offered by U-Blox that can guarantee a precision up to a few centimeters. In PointPerfect and in similar services, augmented information can be broadcast via satellite, and it is encrypted in order to limit access only to paid users. Legitimate receivers are assumed to enjoy a secure and confidential Internet connection, which is infrequently used by the service provider to distribute the decryption keys. For example PointPerfect requires customers to connect to Internet once a month. The current version of the SPARTN encryption, including the Dynamic Key message to distribute securely the decryption key, is not optimized to protect the service provider against piracy when the number of users increases above a certain threshold defined by the satellite available bandwidth. Thus, each service provider needs to put in place mechanisms to protect against it in order to avoid that pirate customers are able to redistribute or resell decryption keys to illegitimate users.

In this paper we propose *APBE* (*Anti-Piracy Broadcast Encryption*), a method to protect premium augmentation data distributed via satellite broadcast, capable of counteracting pirates in a bandwidth-saving and storage-saving way. APBE requires only a small and constant bandwidth overhead over the satellite link, and a small and constant storage space and CPU time on the receivers. Also, APBE is a candidate for the message protection in the next SPARTN format specifications.

The rest of the paper is organized as follows. Section 2 discusses and compares with relevant related work. Section 3 summarizes the SPARTN format as specified by the current

[1] https://www.u-blox.com/en/blogs/insights/how-scalable-high-precision-gps-resetting-expectations-five-markets

[2] https://www.spartnformat.org/

[3] https://www.u-blox.com/en/product/pointperfect

version of the standard. Section 4 introduces our reference adversary model. Section 5 describes the proposed APBE system in detail. Section 6 analyzes its security against the reference adversary model. Section 7 shows and discusses the results of our experimental evaluation of APBE. Finally, the paper is concluded in Section 8.

2 Related Work

The SPARTN standard is thought for GNSS augmentation systems, which provide receivers with augmentation data aimed at computing their position quicker and more accurately. SPARTN is a PPP-RTK GNSS augmentation format. PPP-RTK is a state space representation solution that differs from the PPP global solutions as it provides ionosphere and troposphere data at local/regional level and at the same time differs from RTK which is instead an observation space representation solution relaying on an high density of reference stations, each covering around 50 kilometers. Other standards exist for GNSS augmentation, the most prominent of which are SBAS, RTCM SSR 10403.3, and LPP.

SBAS (Satellite-Based Augmentation System) [12] is standard defined by ICAO, which transmits augmentation data through geostationary satellites. SBAS is addressed mainly to improving the integrity of the positioning service in civil aviation applications. With respect to SPARTN, SBAS does not provide for local ionosphere and troposphere corrections, and it provides for the vertical ionospheric delays in meters at the grid points located at every five by five degrees in latitude and longitude. RTCM SSR 10403.3 (Radio Technical Commission for Maritime Services, State Space Representation) [9] is a standard defined by RTCM to represent augmentation data, which can be transmitted via several different links, for example satellite or the Internet. RTCM SSR 10403.3 does not define any cryptographic service, so it must be equipped with some layer-link encryption in order to protect data. None of the above standards provides for piracy protection, and neither they embed cryptography to protect augmentation data at all. They typically demand commercial data protection to lower layers. We note however that our bandwidth-saving anti-piracy technique could be fruitfully adapted to broadcast augmentation standards like SBAS and RTCM SSR 10403.3, of course assuming that receivers can (infrequently) establish secure Internet connections to receive the decryption keys, or that they are securely preloaded with the decryption keys.

LPP (LTE Positioning Protocol)[14] is a standard defined by 3GPP to provide augmentation data via cellular communication. Augmentation data can be transmitted unicast or broadcast, and they can be encrypted. Moreover, as long as SIM cards remain uncompromised, customers cannot resell decryption keys to illegitimate users, so in practice LPP avoids piracy thanks to hardware. However, LPP forces receivers to be equipped with cellular transceivers and SIM cards, and to pay a mobile network operator on top of the GNSS augmentation service provider.

While the broadcast encryption techniques date back to 1993[7], the first broadcast encryption scheme having performance characteristics good enough to be applied in satellite IoT applications was published by Boneh, Gentry and Waters in 2005[3]. From now on, we will refer to such a scheme as "BGW05". BGW05 has both ciphertext size and private key size constant with respect to the number of receivers, therefore it allows for a low overhead on the satellite channel and a low traffic for the distribution of private keys. Unfortunately, in BGW05 the receivers need also the public key in order to decrypt, and the public key size is linear with the number of receivers. Therefore, BGW05 is not suitable for IoT applications in which the receivers have a limited amount of memory. In the same paper, the authors present also a technique to reduce the size of the public key, but this comes at the cost of sensibly increasing the size of the ciphertext, which is unacceptable for a satellite communication in which bandwidth comes at premium. In this paper, we employ a modified version of BGW05, which allows us to occupy a small and constant amount of memory on the receivers, at the cost of losing the capability of selecting the receivers able to decrypt, which is not needed for our GNSS augmentation application.

A technique strictly correlated to broadcast encryption is traitor-tracing broadcast encryption, which aims at preventing receivers to provide for decryption algorithms or decryption devices without being identified. Although traitor-tracing schemes are explicitly addressed to combat online piracy, we preferred not to use them for our anti-piracy system. The reason is that they generally consume much more bandwidth than BGW05. The most bandwidth-saving traitor-tracing scheme in the literature is the one by Guillot et al.[8], whose ciphertext is six group elements instead of two of APBE.

3 SPARTN

SPARTN[11, 6] is an open industry standard thought for Point Precision Positioning Real-Time Kinematic (PPP-RTK) GNSS augmentation systems, which provide receivers with augmentation data aimed at computing their position quicker and more accurately. The SPARTN format provides for various types of data messages, conveying orbit, clock, bias, and atmosphere corrections by geographic area. SPARTN supports several GNSS constellations, like GPS, Galileo, BeiDou, etc. SPARTN has been designed for flexibility, robustness and future evolution. In SPARTN there are different orbit, clock, and bias flexible messages per GNSS constellation and different high-precision (iono- and tropo-) atmosphere correction messages per GNSS constellation for each area defined by the geographic area definition message. The SPARTN format allows the correlation across messages via a specific "issue of update" and it provides multiple continuity indicators.

In the present paper, we are mainly focused on the SPARTN encryption capabilities. The SPARTN 2.0.1 format allows the messages to be encrypted with a 128-bit symmetric key (called *data encryption key* in the present paper) by means of AES-CTR. Receivers are assumed to obtain the data decryption key through a secure secondary channel, which is not the focus of the present paper. They could use sporadic Internet connections or simply have all the lifetime data encryption keys preloaded in a root-of-trust component. For example PointPerfect requires customers to connect to

Internet once a month and establish a secure MQTT session with the service provider. Whatever the secure secondary channel is, we assume that receivers cannot be *always online*. Otherwise, they could receive the GNSS augmentation data directly via the Internet, without leveraging the satellite link.

4 Adversary Model

We consider two kinds of adversary: the *eavesdropper* and the *pirate*. The eavesdropper is someone that wants to read the GNSS augmentation data without paying. As long as the data decryption key is securely distributed only to the paying customers, the eavesdropper adversary is already thwarted by the current SPARTN 2.0.1 standard. We consider this adversary in order to maintain the same security guarantees in APBE.

On the other hand, the pirate is a customer or a group of colluding customers that want to disclose their keys to other users to make the augmentation service illegitimately available to them with a fee. The pirate also wants to remain anonymous, so that the service provider cannot identify her[4] and remove her from the SPARTN network. As we did with legitimate customers, we assume that the pirate's customers cannot be always online. Otherwise, they could receive the decrypted GNSS augmentation data from the pirate simply via Internet. In this case, there would be no possibility to protect the legitimate service provider by cryptography. We also assume that the service provider can eventually detect piracy attacks, for instance by monitoring activities on the dark web, and he can recover the key that the pirate distributed to her illegitimate customers. This means that the pirate's activity does not remain concealed forever, otherwise we could not take actions against her. It means also that the pirate does not sell her key inside tamper-proof hardware decoders. Doing this is of course possible, but it would cause the pirate service to be quite expensive, and thus hardly competitive with respect to the legitimate one. There are in the literature some solutions to identify pirates that sell tamper-proof hardware, but this comes with an important increase in bandwidth consumption[2].

With the current SPARTN 2.0.1 format explained in Section 3, the piracy attack is shown in Figure 1. The pirate receives the data encryption key and forwards it to her customers, which can therefore decrypt the augmentation data without paying the legitimate service provider. In order to remain anonymous, the pirate can use onion routing to distribute the data encryption key to her clients (for example by means of a dark-web site), and bitcoins to receive possible payments. There is nothing in the disclosed data encryption key which identifies the pirate, since the data encryption key at each given time is unique for the whole system.

5 Anti-Piracy Broadcast Encryption

Figure 2 shows the general architecture of APBE. The *users* are entities willing to receive premium augmented GNSS data. Each user controls one or more *receivers*, which are devices capable of receiving messages from the satellite link. We assume that the receivers are quite constrained in

[4]From now on, we will refer to the adversary as "she", reconnecting with the tradition in cryptography that names an adversary "Eve" or "Mallory".

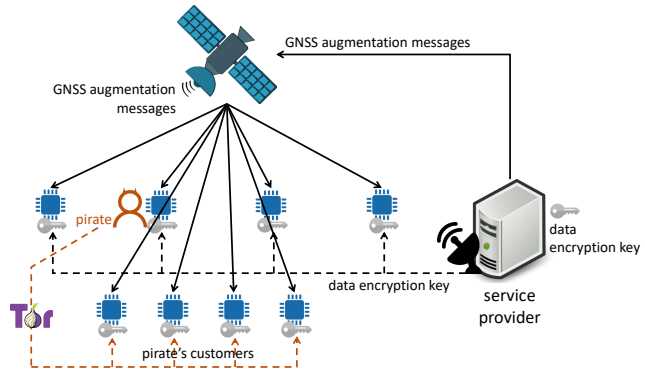

Figure 1. Piracy attack against SPARTN 2.0.1. Solid lines represent satellite links, dashed lines represent the secure secondary channel, red dashed lines represent the anonymous channel by which the pirate communicates with her illegitimate customers.

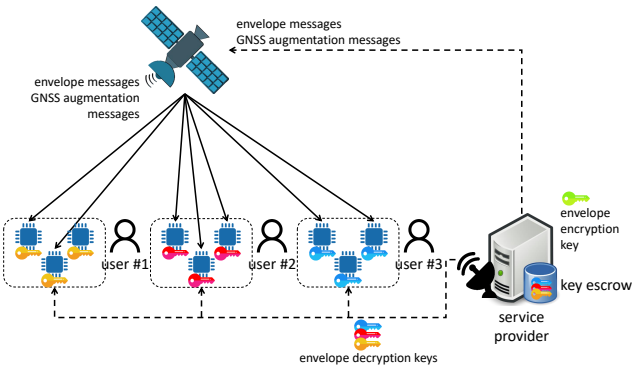

Figure 2. APBE architecture

terms of storage capacity and processing power. The *service provider* is a node in charge of preparing and encrypting the SPARTN messages and delivering them to the satellite for the broadcast. It is also in charge of creating and distributing the private keys to the receivers by means of a secure secondary channel. In contrast to the SPARTN 2.0.1 format, every user is associated to a *different* key, which he distributes to all the receivers controlled by him. Such a key is not the data encryption key, but rather an *envelope decryption key* (the meaning of this name will be clear afterwards). The service provider keeps track of which envelope decryption key has been given to which user by means of a database (*key escrow*), in order to provide for piracy accountability. As soon as a pirate discloses her envelope decryption key to her illegitimate customers, the service provider can identify her and undertake the proper actions.

The data encryption key is still used, but the service provider frequently changes it by transmitting it on the satellite link in an encrypted form, in a message that we call *envelope message*. The data encryption key is encrypted by means of a broadcast encryption scheme based on bilinear pairings[5]. Such a broadcast encryption scheme allows the service provider to send a *single* envelope message for all the receivers, in such a way to consume satellite bandwidth in a scalable manner. The envelope messages are en-

crypted by the service provider with an *envelope encryption key*, and decrypted by each receiver with its own envelope decryption key. The broadcast encryption scheme includes a `Setup` algorithm, an `Encrypt` algorithm, and a `Decrypt` algorithm. The `Setup` algorithm is sporadically run by the service provider, each time it wants to distribute new envelope decryption keys to all its customers. It creates an envelope encryption key and a set of envelope decryption keys, one for each user. The envelope encryption key is kept secret by the service provider, whereas the envelope decryption keys are confidentially distributed to the users through the secure secondary channel. Note that it is not possible for the service provider to create a new envelope decryption key without running again the `Setup` algorithm and thus changing all the existing envelope decryption keys. For this reason, it could be convenient for the service provider to create more envelope decryption keys than customers, in order to accommodate possible new customers in a flexible way. The `Encrypt` algorithm is run by the service provider to generate a new random data encryption key and to encrypt it with the envelope encryption key, thus producing an envelope message. The `Decryption` algorithm is run by the receivers to decrypt an envelope message and extract a data encryption key, which in turn they use to decrypt the actual augmentation data. The service provider sends envelope messages with high frequency, for example each 30 seconds or 1 minute, thus frequently changing the data encryption key.

The broadcast encryption scheme is parameterized with a pairing-friendly elliptic curve of Type III, basing on which it is possible to define: (i) an additive cyclic group $\mathbb{G}_1$ of curve points, with generator P_1 and order r; (ii) an additive cyclic group $\mathbb{G}_2$ with generator P_2 and same order r; (iii) a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$, where $\mathbb{G}_T$ is a multiplicative cyclic group with generator $e(P_1,P_2)$ and same order r; (iv) a hash function $H : \mathbb{G}_T \longrightarrow \{0,1\}^k$, where k is the bit length of the data encryption keys. We assume the above parameters are known by the scheme algorithms. The scheme algorithms are defined as follows.

- `Setup`. It takes as input the number of customers n, and it outputs n envelope decryption keys EDK_i with $i \in [1 \dots n]$ and an envelope encryption key EEK. The algorithm first picks two random numbers $\alpha, \gamma \in \mathbb{Z}_r$, and then it computes the envelope encryption key as:

$$EEK = \gamma[P_1] + \sum_{j \in [1 \dots n]} \alpha^j[P_1], \tag{1}$$

  and the envelope decryption key for each user i as:

$$EDK_i = \left\{ Y_i = \alpha^i[P_2], K_i \right\}, \tag{2}$$

  where:

$$K_i = \gamma\alpha^i[P_2] + \sum_{j \in [1 \dots n], j+i \neq n+1} \alpha^{j+i}[P_2]. \tag{3}$$

- `Encrypt`. It takes as input the envelope encryption key, and it outputs a new random data encryption key and an envelope message. The algorithm first picks a random $t \in \mathbb{Z}_r$, and then it computes $C_0 = t[P_1]$ and $C_1 = t[EEK]$. The data encryption key will be:

$$DEK = H\left(e(P_1,P_2)^{t\alpha^{n+1}}\right), \tag{4}$$

  and the envelope message will be:

$$C = \{C_0, C_1\}. \tag{5}$$

- `Decrypt`. It takes as input an envelope message and an envelope decryption key, and it outputs the data encryption key. Supposing the algorithm is run by a receiver of user i, it recovers the data encryption key as follows:

$$\begin{aligned} DEK &= H(e(C_1,Y_i)/e(C_0,K_i)) \\ &= H\left(\frac{e\left((\gamma+\alpha^1+\dots+\alpha^n)[P_1], \alpha^i P_2\right)}{e\left(t[P_1], (\gamma\alpha^i+\alpha^{i+1}+\dots+\alpha^{i+n}-\alpha^{n+1})[P_2]\right)}\right) \\ &= H\left(e(P_1,P_2)^{t\left(\gamma+\alpha^1+\dots+\alpha^n\right)\alpha^i - t\left(\alpha^i(\gamma+\alpha^1+\dots+\alpha^n)-\alpha^{n+1}\right)}\right) \\ &= H\left(e(P_1,P_2)^{t\alpha^{n+1}}\right). \end{aligned} \tag{6}$$

The above broadcast encryption scheme has been derived from the BGW05 one[3], with three modifications. First, we fixed the set of receivers of each broadcast to be always the whole set of receivers in the system. This allows us to avoid storing the entire public key of BGW05 on the receivers, so that to obtain a constant and minimal storage consumption on the receivers. The whole public key (that is represented by the envelope encryption key in this paper) is kept only by the service provider. Second, we provide for a single broadcaster, which we trust not to disclose cryptographic quantities. This does not change anything in the scheme, but it allows us to keep the envelope encryption key secret, so to avoid a particular piracy attack against BGW05 (see Section 6). Third, we translated the BGW05 scheme from the Type-I pairing setting (in which $\mathbb{G}_1 = \mathbb{G}_2$) into the Type-III pairing setting (in which $\mathbb{G}_1$ and $\mathbb{G}_2$ are different groups, and $\mathbb{G}_1$ elements have smaller size than $\mathbb{G}_2$ elements). This leads to sensibly smaller $\mathbb{G}_1$ elements with the same security level, which in turn makes the envelope messages smaller, and the decryption algorithm quicker. Therefore, we both save bandwidth on the satellite link and CPU resources on the constrained receivers.

6 Security Analysis

Since the APBE broadcast encryption scheme is a specific case of the BGW05 scheme, it inherits its security proof against the eavesdropper adversary. More precisely, it guarantees semantic security against a chosen-plaintext attacker that does not have a valid envelope decryption key.

Regarding the pirate adversary, if the pirate simply discloses her envelope decryption key to her customers she suddenly loses her anonymity in front of the service provider, which can identify her by means of the key escrow. Alternatively, the pirate can disclose the data encryption key after having decrypted it, which is unique for all the system, so it is not tied to a particular user. However, as the service provider changes the data encryption frequently (e.g., once each 30 seconds), the pirate's customers should be always online to receive new data encryption keys, which is unacceptable for them.

As a final option, the pirate (or a colluding set of pirates) could "mix" somehow their envelope decryption keys to build an anonymous pirate decoder. Since we employ a broadcast encryption scheme without traitor tracing guarantees, we do not have a mathematical proof that there exists no such a method. However, we strongly believe that no such method exists, for the reasons we will explain below. From the year of its appearance (2005, so 18 years at the time of writing), no untraceable piracy attack against BGW05 has been published, with only one exception. Such an exception is represented by an interesting attack by Weng et al. in 2007 [13]. In such an attack, two or more pirates collude and mix their decryption keys to compute the parameters of an algorithm that is able to decrypt messages. The authors give a formal proof that, by knowing the pirate decryption algorithm and its parameters, the service provider cannot identify any pirate of the colluding ones. Although the untraceable piracy attack by Weng et al. is ingenious, it cannot be applied to our modified version of BGW05. This is because we do not publish the public key, which in our scheme is represented by the envelope encryption key. In particular, the attack requires to know specific elements of the envelope encryption key, which are not recoverable from the pirates' envelope decryption keys, unless solving an elliptic-curve discrete logarithm. In other words, by keeping secret the envelope encryption key, we lose the possibility of having multiple service providers that generate augmentation messages (possibility that BGW05 has), but at the same time we defend against untraceable piracy attacks in literature.

7 Performance Evaluation

In order to demonstrate the feasibility of the proposed scheme, the APBE scheme has been implemented in a proof of concept deployment. The deployment comprises of a service provider prototype, which implements the `Setup` algorithm and the `Encrypt` algorithm, and a receiver prototype, which implements the `Decryption` algorithm. We run the receiver prototype on an ESP32 IoT platform[5] from Espressif, as it is a rapid prototyping platform very popular for its low cost. The board supports FreeRTOS, a popular operating system for microcontrollers. The APBE scheme has been implemented using the RELIC library [1], a cryptographic library that implements many of the operations required by the APBE scheme and different pairing-friendly elliptic curves.

The APBE implementation is used to run a set of experiments to assess its performance on a realistic scenario. For every experiment the service provider executes the `Encrypt` algorithm once, and then the envelope message is transmitted to the receiver, which executes the `Decrypt` algorithm once. Four different curves are considered in our experiments to analyze different options that trade-off security versus complexity, namely BN_P158 (80-bit security), BN_P254 (100-bit security), IETF-standardized BN_P256 (100-bit security) and BLS12_381 (128-bit security). The following metrics are measured on each experiment:

- *Envelope size*, defined as the overall size of the envelope message produced by the service provider in bits. This gives a measure of the bandwidth overhead consumed by APBE on the satellite link.
- *Initialization time*, defined as the time required by the receiver to perform all the initialization procedures required for envelope decryption. This set of operations include the initialization of the library and the parameters of the elliptic curve. These operations are executed only once at the receiver's bootstrap.
- *Decryption time*, defined as the time between the reception of the last bit of the envelope message and the completion of the decryption process on the receiver.
- *Max stack size*, defined as the peak stack occupation during decryption in bytes. This metric estimates the storage required on receivers for executing decryptions.

[5] `https://www.espressif.com/en/products/socs/esp32`

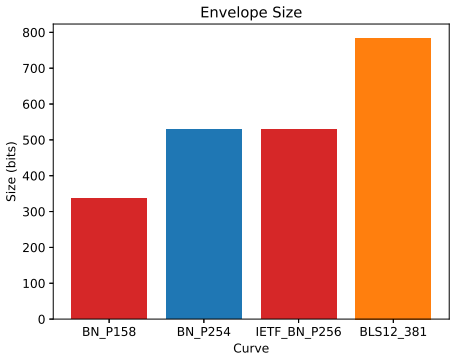

Figure 3. Envelope size with different elliptic curves

To obtain statistically sound results, 200 independent replicas are run for each scenario, average results are reported. The 95th percentile confidence intervals are also reported, however, they are not visible considering that the intervals are very narrow, due to the low variability of the different replicas.

Figure 3 shows the envelope size resulting from encryption at the service provider with the different elliptic curves considered. As expected, a higher level of security results in a larger envelope size, e.g., BN_P158 with 80-bit security results in an envelope size of approximately 350 bits, while the BLS12_381 with 128-bit security results in almost 800 bits.

Figure 4 reports the initialization time required to initialize the library and the decryption data structures. Also in this case the difference in the results obtained with different curves is noticeable. The difference is marked especially if the curve with the highest security is considered, i.e., BLS12_381. It is worth to highlight that, although the time required for this operation is in the order of seconds, especially for the curves with highest security, this operation must be performed only once at bootstrap by the receiver.

In Figure 5 we show the decryption time for a single envelope with the different curves considered. Also in this case, different curves result in a different decryption time: as the security of the curve increases the decryption time increases as well. This can be explained considering that every decryption operation requires two pairings, whose complexity depends on the sizes of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. If we analyze the

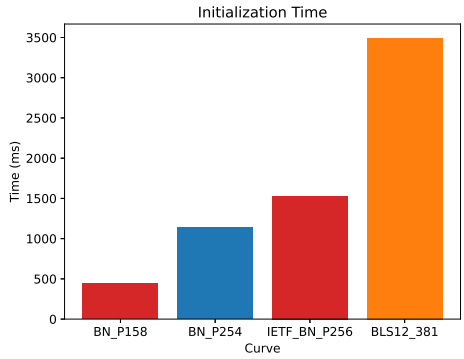

Figure 4. Initialization time with different elliptic curves

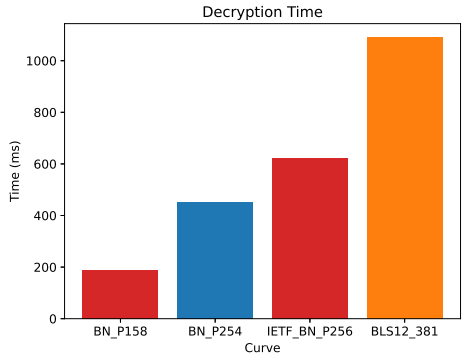

Figure 5. Decryption time with different elliptic curves

decryption time in absolute terms, however, we can notice that the overall time is below 1 second, except for the curve with the highest security, which, however, results in a decryption time of approximately 1 second. Considering a service provider that sends an envelope message every 30 seconds, the receivers will stay busy on decryption operations only $< 1/30$ of their time. This confirms the feasibility of the implementation of the proposed scheme in a real system.

To conclude, in Table 1 we report the average maximum stack size obtained over the 200 repetitions of each scenario. The max stack size has been estimated by enabling the stack watermarking functionality implemented by FreeRTOS. Also in this case, the maximum stack size depends on the complexity of the curve: the higher the security level is, the higher the stack occupation. A deeper analysis highlighted that the majority of the stack occupation is due to the two pairings and the division in $\mathbb{G}_T$, thus explaining the dependence of the stack occupation on the complexity of the curve. Such results, however, confirm that the implementation of the proposed method is suitable for a receiver with a limited RAM capacity. If we analyze the results in absolute terms we can notice that the IETF BN_P256 curve results in a higher stack occupation than the BLS12_P381 curve. This can be explained considering that the former has a higher order r than the latter.

8 Conclusions

In this paper we proposed APBE (Anti-Piracy Broadcast Encryption), a method to protect premium augmentation data distributed via satellite broadcast, capable of counteracting pirates in a bandwidth-saving and storage-saving way. APBE requires only a small and constant bandwidth overhead over the satellite link, and a small and constant storage space and CPU time on the receivers. Also, APBE is a candidate for the message protection in the next SPARTN format specifications.

Table 1. Max stack size in bytes

Curve	Max stack size (bytes)
BN_P158	9100
BN_P254	12500
IETF BN_P256	19600
BLS12_P381	19200

9 References

[1] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

[2] D. Boneh and M. Franklin. An efficient public key traitor tracing scheme. In M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 338–353, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[3] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 258–275, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[4] L. Chen, F. Zheng, X. Gong, and X. Jiang. Gnss high-precision augmentation for autonomous vehicles: Requirements, solution, and technical challenges. *Remote Sensing*, 15(6):1623, 2023.

[5] N. El Mrabet and M. Joye. *Guide to pairing-based cryptography*. CRC Press, 2017.

[6] K. Ferguson, L. Urquhart, and R. Leandro. SPARTN: The first open GNSS data standard that enables safe and accurate GNSS localization for automotive applications. In *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, pages 2092–2106, 2020.

[7] A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 480–491, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[8] P. Guillot, A. Nimour, D. H. Phan, and V. C. Trinh. Optimal public key traitor tracing scheme in non-black box model. In A. Youssef, A. Nitaj, and A. E. Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, pages 140–155, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[9] R. M. Kalafus and K. Van Dierendonck. The new RTCM SC-104 standard for differential and RTK GNSS broadcasts. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 741–747, 2003.

[10] E. D. Kaplan and C. Hegarty. *Understanding GPS/GNSS: principles and applications*. Artech house, 2017.

[11] S. Vana, J. Aggrey, S. Bisnath, R. Leandro, L. Urquhart, and P. Gonzalez. Analysis of GNSS correction data standards for the automotive market. *Navigation*, 66(3):577–592, 2019.

[12] T. Walter. *Satellite Based Augmentation Systems*, pages 339–361. Springer International Publishing, Cham, 2017.

[13] J. Weng, S. Liu, and K. Chen. Pirate decoder for the broadcast encryption schemes from crypto 2005. *Science in China Series F: Information Sciences*, 50(3):318–323, Jun 2007.

[14] C. Yang, S. Mao, and X. Wang. An overview of 3GPP positioning standards. *GetMobile: Mobile Computing and Communications*, 26(1):9–13, 2022.