Secure Communication via GNSS-based Key Synchronization

Maki Yoshida[1], Sumio Morioka[2] and Satoshi Obana[3]

[1]National Institute of Information and Communications Technology, Tokyo, Japan
[2]Interstellar Technologies Inc., Tokyo, Japan
[3]Hosei University, Tokyo, Japan

Abstract

In this paper, we show that accurate GNSS timing information contributes to the realization of highly secure communication. Specifically, we first present a cryptographic protocol that prevents not only spoofing and eavesdropping but also replay attack by synchronizing cryptographic keys based on GNSS time and estimated latency. Compared with classical cryptographic protocols used in the Internet, the proposed protocol is more suitable for a space flight environment in the sense that neither interaction nor state information (data stored in volatile memory such as counter) is required for key synchronization.

Keywords

GNSS, secure communication, spoofing, replay attack, confidentiality and integrity, spacecraft.

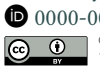
1. Introduction

Global Navigation Satellite System (GNSS) such as Galileo [1, 2] has made significant progress in recent years so that more accurate and reliable positioning, navigation and timing information can be available. The availability of accurate and reliable GNSS timing information is becoming essentially important in order to establish highly secure communication. Specifically, cryptographic keys are frequently updated and evolved according to time (updatable/evolving cryptography with forward security and/or post-compromised security [3], end-to-end encryption such as Zoom [4]).

A common approach to synchronize cryptographic keys in the previous work is for the sender and receiver to share state information mapped to a key index, interact with each other, and update the state. This approach is reasonable for communication systems over a classical channel such as TLS over the Internet and WPA3 over a wireless LAN. However, for a space flight environment, it is vulnerable to deadlocks due to interactions over a noisy channel and soft errors destroying state information. More specifically, key synchronization based on state information may cause permanent loss of communication since key is no longer synchronized if state information shared between the sender and receiver is not identical. Such accident is much more likely in a space flight environment than usual communication. The general

WIPHAL 2023: Work-in-Progress in Hardware and Software for Location Computation, June 06–08, 2023, Castellon, Spain

maki-yos@nict.go.jp (M. Yoshida); sumio.morioka@istellartech.com (S. Morioka); obana@hosei.ac.jp (S. Obana)
0000-0002-1267-0058 (M. Yoshida); 0000-0001-7641-1904 (S. Morioka); 0000-0003-4795-4779 (S. Obana)



solution for mitigating the risk of communication loss caused by channel/soft error is to employ error correcting code. However, employing error correcting code for mitigating soft error is insufficient in the following sense. Firstly, it is difficult to precisely predict the pattern and/or amount of error occuring during space flight. Secondly, the damage caused by soft error is enormous once error correcting code fails to correct errors or decodes incorrectly.

The objective of this paper is the development of secure two-party communication that synchronizes keys without interaction and without maintaining state information associated with keys, which is mainly designed for a space flight environment. Our method is to use GNSS timing information at both sender and receiver sides and estimate a possible latency of the involved sub-systems.

First, we define a system model for secure two-party communication via GNSS-based key synchronization. We focus on one-way communication in order to mitigate the possibility of falling into deadlock due to interaction on an insecure channel. We also define a security model against eavesdropping, spoofing, and replay attacks in a major cryptographic manner so-called "attack game." Second, we propose a protocol that realize highly secure communication via GNSS-based key synchronization. The basic idea is to estimate latency of sub-systems based on three layers and channel. The security of the proposed protocol is cryptographically proven under an assumption that the protocol uses secure a simple building block called authenticated encryption with associated data.

From our results, higher accuracy of GNSS time and estimated latency/error can lead both higher security. In addition, it is important to design a communication system so that the latency of involved sub-systems is guaranteed as much as possible (for example, employing hardware implementation as much as the cost allows). In other words, a requirement on latency is useful for determining a system implementation policy.

2. System Model

We define a system model of secure communication via GNSS-based time synchronization, denoted by $\mathsf{SC}_{\mathsf{GNSS}}$, where a sender S and a receiver R aim to establish security (confidentiality and integrity) over an insecure communication channel $\mathcal{C}$ based on GNSS and cryptography by pre-sharing a finite number of cryptographic keys that are indexed by "epoch". Let N be the maximum number of used keys (or possible epochs) in the system lifetime and k_τ be the key indexed by epoch τ. Each epoch τ is derived by some GNSS time t by an index function IND so that $\tau = \mathsf{IND}(t)$. The keys at sender side (resp. receiver side) are stored in a key storage denoted by $KS^{(S)}$ (resp. $KS^{(R)}$).

Our system model follows the Shannon-Weaver model in [5, 6] consisting of five basic components: a source and a transmitter at S, a channel, a receiver and a destination at R. In the Shannon-Weaver model, the source produces the original message. The transmitter translates the message into a signal which is sent using a channel. The receiver translates the signal back into the original message and makes it available to the destination. For example, in communication from a ground station to a spacecraft, the ground station and the spacecraft are S and R, respectively. The source and the destination are their control devices that outputs and receives payloads such as commands, respectively. Then, the ground station and the spacecraft

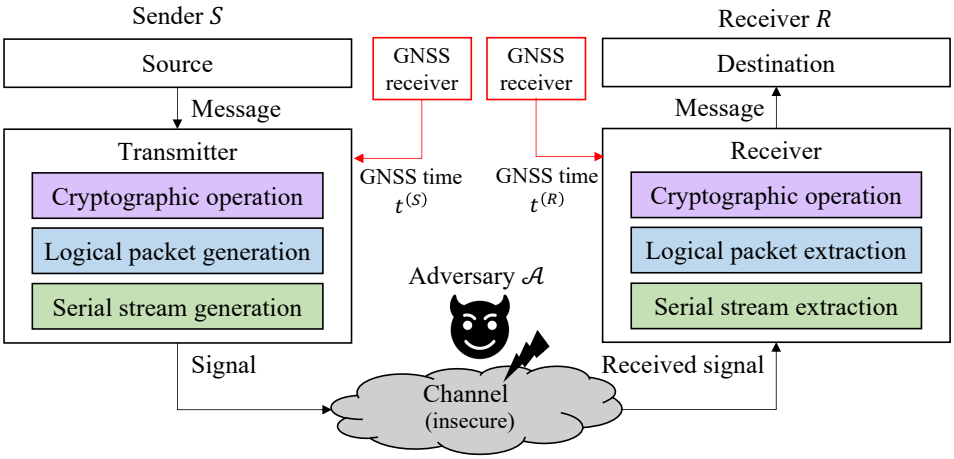


Figure 1: The system model.

uses their hardware/software as a transmitter and a receiver in order to operate messages and signals, respectively.

The sender S first obtains a GNSS time $t^{(S)}$ and generates a cryptographic data by executing a cryptographic operation on confidentiality and integrity for a message m and a key $k_{\mathsf{IND}(t^{(S)})}$, then produces a logical packet, and finally produces a serial stream as a signal that is sent on a channel. The channel models both wire and wireless communication. The receiver R continuously tries to extract a serial stream and logical packets. It estimates a GNSS time $t'^{(S)}$ from his GNSS time $t^{(R)}$ and the logical packet and then executes a cryptographic operation for the logical packet with a key $k_{\mathsf{IND}(t'^{(S)})}$. The receiver finally outputs a message or "$\perp$" which mean accept and reject, respectively. The overview of the system model is shown in Figure 1. A system is said to be correct if the original message is output by the receiver when the signal is neither lost nor tampered.

In general, confidentiality and integrity are defined against an adversary who accesses to the above channel by using a game-based technique. We here show an essence of our definition. An adversary $\mathcal{A}$ tries a game where $\mathcal{A}$ can make a sender S to send a signal for any message m at any GNSS time t; the adversary controls the channel in the sense that $\mathcal{A}$ can eavesdrop on, intercept, and tamper any data sent through $\mathcal{C}$, and further can sent any data that $\mathcal{A}$ computes through $\mathcal{C}$ at any time (which includes a replay attack); he can then know whether the receiver accepts received signal or not; he finally outputs 0 or 1. Let $(t_1^{(S)}, m_1^{(S)}), (t_2^{(S)}, m_2^{(S)}), \ldots, (t_p^{(S)}, m_p^{(S)})$ be a sequence of a GNSS time and a sent message at the sender S where $t_i < t_{i+1}$ for any i with $1 \leq i \leq p-1$. Let $(t_1^{(R)}, m_1^{(R)}), (t_2^{(R)}, m_2^{(R)}), \ldots, (t_q^{(R)}, m_q^{(R)})$ be a sequence of a GNSS time and a message that the receiver accepts. In a game for integrity, the adversary is said to lose the game if the following is satisfied: any tuple $(t_j^{(R)}, m_j^{(R)})$ with $1 \leq j \leq q$ is contained in the sender sequence and the order of the receiver tuples are the same as the corresponding sender tuples. The advantage of $\mathcal{A}$ in the integrity game is evaluated by the probability that $\mathcal{A}$ does not lose (or wins). In contrast, in a game for confidentiality, for some i, $\mathcal{A}$ chooses and sends a message $m_{i,1} = m^*$ in addition to $m_{i,0} = m_i^{(S)}$, the sender encrypts either of the messages

$m_{i,b'}$ with a random $b' \in \{0,1\}$. The advantage of $\mathcal{A}$ in the confidentiality game is defined by $|\Pr[b = b'] - 1/2|$, that is, the probability of guessing which message is chosen.

A system is information-theoretically secure (resp. computationally secure) if the advantage of any unbounded (resp. polynomial-time) adversary in both integrity and confidentiality game is negligible. If a system is secure in this sense, then the system does not leak any information on messages and prevents impersonation, forgery, and replay attacks.

We point out that the integrity check is a powerful tool to check the validity of a logical packet because any adversary cannot forge a logical packet. This means that an integrity part or all of the cryptographic operation can be merged into a logical packet extraction. In any case, a GNSS time is estimated before the cryptographic operation to read a key from the key storage.

3. Proposed Secure Communication

In this section, we present a protocol for secure communication via GNSS-based key synchronization. Here, we will give an overview of the proposed protocol for secure communication. We employ secure AEAD for achieving both confidentiality and integrity. For key synchronization between S and R, the proposed protocol employ GNSS. Namely, the transmitter computes an authenticated ciphertext $(t^{(S)}, c, a)$ of m using the key $k := k_{\mathsf{IND}(t^{(S)})}$ read from $KS^{(S)}$ where $t^{(S)}, c, a$ denote time information (this part is not encrypted but protected to ensure integrity), an encryption of m, and authentication tag for $(t^{(S)}, c)$, respectively. The authenticated ciphertext is then given to a logical packet generation. Then, the receiver extracts an authenticated ciphertext $(t^{(S)}, c, a)$ by executing the serial extraction and logical packet extraction, and if $t^{(R)}$ is "consistent" with $t^{(S)}$ (detailed condition is explained in in Section 3.1), then the receiver decrypts/verifies $(t^{(S)}, c, a)$ with $k_{\mathsf{IND}(t^{(S)})}$ to obtain the message m.

The crucial point is to verify the consistency of the time information $t^{(S)}$. We evaluate factors affecting the time difference between $t^{(S)}$ and $t^{(R)}$ in Section 3.1, and assume the existence of its upper/lower bounds (say $\delta^{\uparrow}$ and $\delta^{\downarrow}$). In this case, the consistency of time information $t^{(S)}$ can be verified only by checking that $\delta^{\downarrow} \leq t^{(R)} - t^{(S)} \leq \delta^{\uparrow}$ holds.

In the proposed protocol, both the transmitter and receiver consist of three layers, and shared memory is available for two adjacent layers (We use the notation $SM_{\mathsf{A,B}}$ to represent the shared memory between layer A and B). The lowest layer (serial stream layer: SS) is responsible for generating/extracting raw serial stream that is to be sent through a channel. The middle layer (logical packet layer: LP) is responsible for generating/extracting logical packets. We should note that a single logical packet may consist of multiple physical packets. The highest layer (cryptographic operation layer: CO) is responsible for invoking cryptographic operations to ensure confidentiality and integrity.

3.1. Estimation on the Time Difference

The flow diagram of proposed protocol is given in Figure 2. In the diagram, there are respectively three arrows for S and R where each arrow corresponds to a layer (i.e., either serial stream layer or logical packet layer or cryptographic operation layer). Color lines in each layer indicate time slots in which some operations are executed within the layer.

Table 1
Factors of subsystems for a receiver to estimate a sender's GNSS time and estimated parameter values of command up-link

	Subsystem	Factor	Notation		
			Variable	Upper	Lower
Sender side	GNSS time	Error	$\epsilon_{\mathrm{S_{time}}}$	$\epsilon^{\uparrow}{}_{\mathrm{S_{time}}}$	$\epsilon^{\downarrow}{}_{\mathrm{S_{time}}}$
	Cryptographic operation	Latency	L_{crys}	$L^{\uparrow}{}_{\mathrm{crys}}$	$L^{\downarrow}{}_{\mathrm{crys}}$
	Logical packet generation	Latency	L_{logg}	$L^{\uparrow}{}_{\mathrm{logg}}$	$L^{\downarrow}{}_{\mathrm{logg}}$
	Serial stream generation	Latency	L_{bitg}	$L^{\uparrow}{}_{\mathrm{bitg}}$	$L^{\downarrow}{}_{\mathrm{bitg}}$
	Channel	Latency	L_{chao}	$L^{\uparrow}{}_{\mathrm{chao}}$	$L^{\downarrow}{}_{\mathrm{chao}}$
Receiver side	Serial stream extraction	Latency	L_{bite}	$L^{\uparrow}{}_{\mathrm{bite}}$	$L^{\downarrow}{}_{\mathrm{bite}}$
	Logical packet extraction	Latency	L_{loge}	$L^{\uparrow}{}_{\mathrm{loge}}$	$L^{\downarrow}{}_{\mathrm{loge}}$
	GNSS time	Error	$\epsilon_{\mathrm{R_{time}}}$	$\epsilon^{\uparrow}{}_{\mathrm{R_{time}}}$	$\epsilon^{\downarrow}{}_{\mathrm{R_{time}}}$
	Cryptographic operation	Latency	L_{cryr}	$L^{\uparrow}{}_{\mathrm{cryr}}$	$L^{\downarrow}{}_{\mathrm{cryr}}$

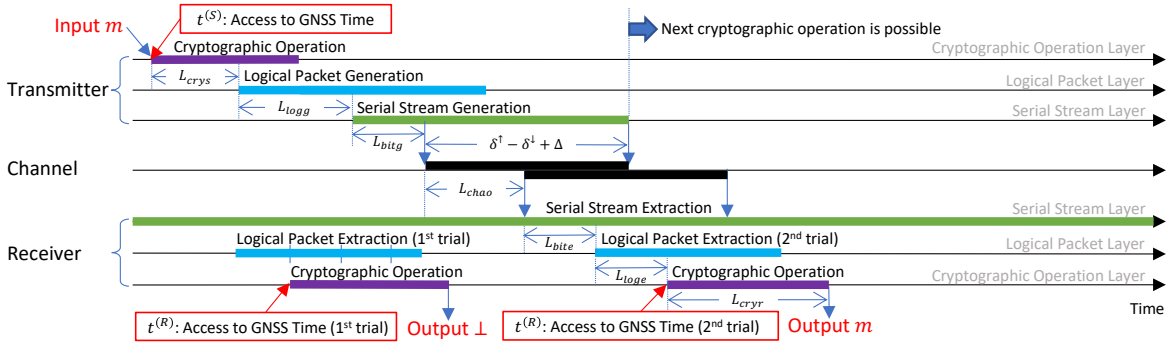


Figure 2: The flow diaglam of secure communication protocol for hardware-oriented receiver.

Table 1 summarises factors affecting the time difference between $t^{(S)}$ and $t^{(R)}$ (time slots corresponding to each factor is indicated by blue double-headed arrow in Figure 2).

GNSS error $\epsilon_{\mathrm{S_{time}}}$ and $\epsilon_{\mathrm{R_{time}}}$ can be estimated by guaranteed value of GNSS receiver device. Each latency factor (e.g., L_{crys} etc.) represents the latency time taken for outputting the first data. Let δ be time difference $t^{(R)} - t^{(S)}$ of the protocol. From Figure 2, δ is estimated as follows.

$$\delta = \epsilon_{\mathrm{S_{time}}} + L_{\mathrm{crys}} + L_{\mathrm{logg}} + L_{\mathrm{bitg}} + L_{\mathrm{chao}} + L_{\mathrm{bite}} + L_{\mathrm{loge}} + \epsilon_{\mathrm{R_{time}}}$$

It is reasonable to assume that there are maximum/minimum value for these factors. Therefore, we can estimate max/min values for δ. Hereafter, we will denote $\delta^{\uparrow}$ to represent maximum value of δ, and denote $\delta^{\downarrow}$ represent minimum value.

3.2. Authenticated Encryption with Associated Data

In the proposed protocol, we employ a cryptographic primitive called authenticated encryption with associated data (AEAD for short) as a building block. AEAD consists of three algorithms $\Pi_{aead} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm Gen takes a security parameter 1^{κ} as input, and outputs a key k. The encryption algorithm Enc takes a key k, a header h, and a

message m with inputs, and outputs an authenticated ciphertext (h, c, a) where c is a ciphertext of m and a is an authentication tag for (h, c). The decryption algorithm takes an authenticated ciphertext (h, c, a) with input, and outputs m or a special symbol $\perp$ where $\perp$ indicates that Dec decides input ciphertext is invalid.

Secure AEAD guarantees both confidentiality and integrity. The formal security notions and their relations among the notions are summarized in [7]. Intuitively, we say that AEAD satisfies IND-CPA (indistinguishability against chosen plaintext attack) (resp. IND-CCA: indistinguishability against chosen ciphertext attack) if it is infeasible to distinguish whether the ciphertext is an encryption of m_0 or m_1 (messages chosen by the adversary) even when the adversary has access to the encryption (resp. decryption) oracle. We also say that AEAD satisfies INT-CTXT (integrity of ciphertexts) if it is infeasible to produce a ciphertext not previously produced by the sender, regardless of whether or not the underlying plaintext is new.

It is shown that AEAD satisfying IND-CPA and INT-CTEXT is constructed by employing Encrypt-then-MAC methodology where underlying symmetric encryption and MAC satisfies IND-CPA and strong unforgeability, respectively [7]. It is also shown that AEAD satisfies both IND-CPA and INT-CTEXT satisfies IND-CCA [7].

3.3. Proposed Protocol

We now present a protocol for secure communication. The protocol mainly targets hardware-oriented receiver in which each layer can operate simultaneously. Let $\Pi_{aead} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be AEAD satisfying IND-CPA and INT-CTEXT. The detailed description of the protocol is described as follows where we assume that each key k_τ $(1 \leq \tau \leq N)$ in the key storage is generated before starting the protocol by $k_\tau \leftarrow \mathsf{Gen}(1^\kappa)$ and shared between S and R.

Transmitter-side algorithm: On input a message m, the Transmitter-side algorithm operates as follows:

Cryptographic Operation:

1. Confirm that neither the logical packet layer nor the serial stream layer is in operation.
2. Get the time information $t^{(S)}$ via GNSS.
3. Read the key $= k_{\mathsf{IND}(t^{(S)})}$ from the storage $KS^{(S)}$ and set $k = k_{\mathsf{IND}(t^{(S)})}$.
4. Encrypt m using AEAD under the key k (i.e., compute $\mathsf{Enc}(k, t^{(S)}, m) = (t^{(S)}, c, a)$)
5. Write $(t^{(S)}, c, a)$ to the shared memory $SM_{\mathsf{CO,LP}}$.

Logical Packet Generation: When the data is written to $SM_{\mathsf{CO,LP}}$, the algorithm constructs a logical packet for an authenticated ciphertext $(t^{(S)}, c, a)$. The resulting logical packet is written to the shared memory $SM_{\mathsf{LP,SS}}$.

Serial Stream Generation: When the packet is written to $SM_{\mathsf{LP,SS}}$, constructs a serial stream for a logical packet and sends the stream to the channel. The important point is that the sender (intentionally or unintentionally) spends $\delta^{\uparrow} - \delta^{\downarrow} + \Delta$ seconds to send out whole serial stream (a time slot corresponding to an upper black line in the channel in Figure 2) where Δ is any positive value. We should note that Δ is inevitable parameter to prevent the adversary from altering the order of message sequence and mounting replay attack. Though smaller Δ is

preferred for better throughput, we may make Δ larger considering, for example, constraint with the underlying channel (e.g., Doppler shift of wireless network).

Receiver-side algorithm: The receiver-side algorithm observes the channel continuously, and when a signal is detected, constructs a logical packet and verifies the integrity of the received message.

Serial Stream Extraction: This layer continuously observes the channel, and writes a serial stream to the shared memory $SM_{\mathsf{SS,LP}}$ when the signal is detected.

Logical Packet Extraction: When the serial stream is written to $SM_{\mathsf{SS,LP}}$, the algorithm scans the serial stream, and extracts the presumed logical packet (the data which looks like a packet but its legitimacy is not clear). The presumed logical packet is written to the shared memory $SM_{\mathsf{LP,CO}}$.

Cryptographic Operation: When the algorithm is in a "wait" state, and the first part of a logical packet is written to $SM_{\mathsf{LP,CO}}$, the algorithm changes the state to "operation", and operates as follows where we use parameter $\delta^{\uparrow}$ and $\delta^{\downarrow}$ estimated in Section 3.1.

1. Get the time information $t^{(R)}$ via GNSS.
2. Extract a presumed authenticated ciphertext $(t'^{(S)}, c', a')$ from the logical packet.
3. Check if $\delta^{\downarrow} \leq t^{(R)} - t'^{(S)} \leq \delta^{\uparrow}$ holds. If it does not, the algorithm outputs $\perp$ and change the state to "wait".
4. Read the key $k_{\mathsf{IND}(t'^{(S)})}$ from the storage $KS^{(R)}$ and set $k' = k_{\mathsf{IND}(t'^{(S)})}$.
5. Outputs $\mathsf{Dec}(k', (t'^{(S)}, c', a'))$ (i.e., if Dec outputs $\perp$ then the authenticated ciphertext is not valid. Otherwise the receiver-side algorithm verifies the legitimacy of the presumed packet), and change the state to "wait".

Here, we show the security of the protocol. The proof of confidentiality is derived from IND-CPA and INT-CTEXT (and, therefore, IND-CCA) security of the underlying AEAD in a straightforward manner, and is omitted here. The integrity of the protocol is shown by the following theorem.

Theorem 1. *Let* $Seq^{(S)} = (t_1^{(S)}, m_1^{(S)}), \ldots, (t_p^{(S)}, m_p^{(S)})$ *be a sequence of a GNSS time and a sent message at* S *where* $t_i < t_{i+1}$ *for any* i *with* $1 \leq i \leq p-1$. *Let* $Seq^{(R)} = (t_1^{(R)}, m_1^{(R)}), \ldots, (t_q^{(R)}, m_q^{(R)})$ *be a sequence of a GNSS time and a message that the receiver accepts in the presence of an adversary* $\mathcal{A}$. *Then no adversary can make the receiver receive* $\mathsf{Seq}^{(R)}$ *such that the adversary wins the game defined in the system model with non-negligible probability.*

Proof: The adversary $\mathcal{A}$ wins the game only if either of the following conditions is satisfied.

1. **Cryptographic spoofing**: There exists $(t_i^{(R)}, m_i^{(R)})$ $(1 \leq i \leq q)$ that is not contained in $Seq^{(S)}$.
2. **Order alteration**: There exist i, j, k, ℓ such that $(t_i^{(S)}, m_i^{(S)}) = (t_k^{(R)}, m_k^{(R)})$ and $(t_j^{(S)}, m_j^{(S)}) = (t_\ell^{(R)}, m_\ell^{(R)})$ and $i < j$ hold but $k > \ell$.
3. **Replay**: There exists i, j, k such that $(t_i^{(S)}, m_i^{(S)}) = (t_j^{(R)}, m_j^{(R)}) = (t_k^{(R)}, m_k^{(R)})$.

The probability that the condition 1) is satisfied is negligible since AEAD used in the protocol satisfies INT-CTEXT. Therefore, all $(t_i^{(R)}, m_i^{(R)})$ are contained in $Seq^{(S)}$ with overwhelming probability. Next, we show that the condition 2) is not satisfied with probability 1. Let $T_i^{(R)}$ and $T_j^{(R)}$ be GNSS times which the receiver obtained in receiving $(t_i^{(S)}, m_i^{(S)})$ and $(t_j^{(S)}, m_j^{(S)})$, respectively. The condition $k > \ell$ is satisfied only if $T_j^{(R)} < T_i^{(R)}$ holds. Since $i < j$ holds and the transmitter spends $\delta^{\uparrow} - \delta^{\downarrow} + \Delta$ seconds to send out whole serial stream corresponding to $(t_i^{(S)}, m_i^{(S)})$, and the next cryptographic operation does not start until the serial stream generation ends, the following inequality must hold.

$$t_j^{(S)} \geq t_i^{(S)} + \delta^{\uparrow} - \delta^{\downarrow} + \Delta \tag{1}$$

Moreover, since the receiver accepts $(t_j^{(S)}, m_j^{(S)})$, the inequality $T_j^{(R)} \geq t_j^{(S)} + \delta^{\downarrow}$ holds. From eq. (1), the inequality is rewritten by $T_j^{(R)} \geq t_i^{(S)} + \delta^{\uparrow} + \Delta$, which proves $T_j^{(R)} > T_i^{(R)}$ since the receiver's acceptance condition of $(t_i^{(S)}, m_i^{(S)})$ implies $t_i^{(S)} + \delta^{\uparrow} \geq T_i^{(R)}$ and $\Delta > 0$ holds.

Finally we show that the condition 3) is not satisfied with probability 1. Let $T_j^{(R)}$ and $T_k^{(R)}$ be GNSS times which the receiver obtained in receiving $(t_j^{(R)}, m_j^{(R)})$ and $(t_k^{(R)}, m_k^{(R)})$, respectively. Without loss of generality we can assume $j < k$ holds. Since the receiver accepts $(t_j^{(R)}, m_j^{(R)})$ the inequality $T_j^{(R)} = t_i^{(S)} + \delta$ must hold where $\delta^{\downarrow} \leq \delta \leq \delta^{\uparrow}$. Moreover, since the transmitter spends $\delta^{\uparrow} - \delta^{\downarrow} + \Delta$ seconds to send out whole ciphertext, $T_k^{(R)} \geq t_i^{(S)} + \delta + (\delta^{\uparrow} - \delta^{\downarrow} + \Delta)$ must hold. We should note that we may adjust the value of Δ depending on the underlying network (e.g., that suffering from Doppler shift). Therefore, $\min_{\delta} T_k^{(R)} = t_i^{(S)} + \delta^{\downarrow} + (\delta^{\uparrow} - \delta^{\downarrow} + \Delta) = t_i^{(S)} + \delta^{\uparrow} + \Delta$ holds. This implies the receiver rejects $(t_i^{(S)}, m_i^{(S)})$ since $T_k^{(R)} - t_i^{(S)} \leq \delta^{\uparrow}$ does not hold. □

4. Feasibility of the Required Performance

In this section, we theoretically examine a possible communication speed under the proposed method. Let $P_{int}^{(S)\downarrow}$ denote the minimum difference of $t_i^{(S)}$ and $t_{i+1}^{(S)}$ for any $i(\geq 1)$, i.e., the minimum interval time of starting the cryptographic operation in the transmitter side[1]. Also, let N_{bitg} and T_{chao} denote a bit count of data sent in the channel and a time to send one bit on the channel, respectively.

The values $P_{int}^{(S)\downarrow}$ and N_{bitg} are given in a requirement to communication system and are not adjustable, while the other parameters are adjustable at implementation stage.

Theorem 2. *For given parameters $P_{int}^{(S)\downarrow}$, N_{bitg}, $\delta^{\uparrow}$ and $\delta^{\downarrow}$, a continuous communication is possible satisfying Theorem 1, if all of the following conditions hold;*
(a)[data length] $N_{bitg} \times T_{chao} \geq \delta^{\uparrow} - \delta^{\downarrow} + \Delta$,
(b)[transmitter operation time] $P_{int}^{(S)\downarrow} \geq L^{\uparrow}{}_{\mathrm{crys}} + L^{\uparrow}{}_{\mathrm{logg}} + L^{\uparrow}{}_{\mathrm{bitg}} + N_{bitg} \times T_{chao}$, and
(c)[receiver operation time] $P_{int}^{(S)\downarrow} \geq L^{\uparrow}{}_{\mathrm{bite}} + L^{\uparrow}{}_{\mathrm{loge}} + L^{\uparrow}{}_{\mathrm{cryr}}$.

Proof: Clear from the parameter definitions and time sequence shown in Figure 2.

[1]In this paper we assume the packet length in serial stream layer is fixed, although we can extend the discussion to variable packet length without difficulty.

5. Conclusion

In this paper, we have proposed a protocol that realizes highly secure communication via GNSS-based key synchronization, which mainly targets space flight environment.

A possible future work is to develop cryptographic protocols resilient to variations of communication environment such as store-and-forward one. Another possible future work is the improvement of throughput. For example, we would like to examine that a short-time internal state can improve the throughput.

References

[1] EUSPA/EC, "Galileo Open Service Navigation Message Authentication (OSNMA) Signal-in-Space (SIS) Interface Control Document (ICD)," Issue 1.0, 2022.

[2] EUSPA/EC, "Galileo Open Service Navigation Message Authentication (OSNMA) Receiver Guidelines," Issue 1.0, 2022.

[3] J. Alwen, S. Coretti, and Y. Dodis, "The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol," EUROCRYPT 2019, LNCS 11476, pp.129–158, 2019.

[4] "Zoom Cryptography Whitepaper," https://github.com/zoom/zoom-e2e-whitepaper (Last accessed on 28th Feb. 2023).

[5] C.E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, vol.27, no.3, p.381, 1948.

[6] C.E. Shannon and W. Weaver, The Mathematical Theory of Communication, University of Illinois Press. ISBN 978-0-252-72546-3, 1998.

[7] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," Journal of Cryptology, vol. 21, no. 4, pp. 469-–491, 2008.