Securing Software Updates under Receiver Radio Frequency Geolocation Risk

Hayden, Blake; Sweeney, Matthew; Hale, Britta

Naval Postgraduate School

Securing Software Updates under Receiver Radio Frequency Geolocation Risk

Blake Hayden and Matthew Sweeney and Britta Hale

{blake.hayden,matthew.sweeney,britta.hale}@nps.edu
Naval Postgraduate School, Monterey, CA

Abstract

In the new, ever-changing cyber domain, it is crucial that the military establishes a method of delivering large data payloads to remote locations that minimizes radio frequency (RF) signature for receivers, thereby reducing the associated geolocation potential. This paper introduces three cryptographic protocols for different components of a delivery architecture for a large data payload from a trusted, back-end source to receivers: a low-response protocol for initial transmission and confirmation and two possible inter-unit distribution protocols with differing optimizations based on connectivity scenarios. All three protocols expressly aim to minimize the radio frequency (RF) footprint created on the receiver end. We provide security models and analyze the security protocols, and furthermore provide a worst-case example bound on RF footprint created at the receivers for each protocol, with variable inputs for data transmission size. These protocols introduce a means for accounting for both security (authentication) and safety (minimized RF footprint) in the delivery of critical data payloads to remote receivers.

1 Introduction

After two decades of counter insurgency operations in urban environments against guerrilla forces, the Marine Corps is shifting focus back to its roots of amphibious warfare. While the Marine Corps is no stranger to the maritime environment, the modern battlefield presents unprecedented challenges to littoral operations. The challenges are amplified by a focus on peer competition which no longer allows the Marine Corps to assume technological superiority over the adversary. In the 38th Commandant's Planning Guidance, General Berger described the shifting focus by saying "the return of great power competition requires a Marine Corps that is primarily organized, trained and equipped for the high-end fight against a peer competitor" [1]. The General goes on to say the Marine Corps needs to be "trained and equipped as a naval expeditionary force-in-readiness and prepared to operate inside actively contested maritime spaces" [1].

To accomplish the Commandant's goal and prepare for the evolving battlefield, the Marine Corps is shifting from its classic force design to new operational focuses such as Distributed Maritime Operations (DMO) and Expeditionary Advanced Basing Operations (EABO). Under the new force structure, operational units will be spread out and separated for extended periods from a headquarters location. Maintaining communications with extended units is vital to the success of the future mission. With the emphasis on communications, comes the critical task of sending software updates to forward units. While forces could rely on returning to headquarters for updates in the past, distributed operations will not have that luxury. Therefore, it is necessary to be able to securely and reliably push updates to subordinate units. Given the information domain capabilities of the enemy, software forwarding capability will be crucial to maintaining the advantage in the information domain.

Situation

With the EABO construct in mind, the protocol must be designed for a headquarters (back-end) unit that sends updates to operating units. The back-end in this scenario will act as the original sender. We operate with the assumption that the back-end has extensive communications equipment and is not in range of adversarial attacks and is therefore not worried about Radio Frequency (RF) footprint. The goal of the sender is to push the update to all receiving units via satellite communications (SATCOM). The message will be sent to multiple units in the same operational area. The job of the receiving units is to receive the update sent via multiple messages, analyze the packets, determine missing packets, confer internally to get all packets, and apply the update.

The goals of the protocol are to authenticate the sender and minimize the RF footprint for the receiver.

Constraints

The main constraint in this problem is the necessity for a low RF signature on the receiving end. Given the dynamic adversarial threat, the receiving units do not have the luxury of communicating with the sender extensively. Any amount of RF footprint emitted by the receiving unit can expose their location to the enemy. Such practices may have worked in the past but will not in an information contested environment. The adversary would easily identify the location of a forward unit if it communicated back to the sender via SATCOM, thus jeopardizing the safety of the unit and success of the mission.

In addition to reducing RF footprint, the transmission needs to provide security against an adversary in whose interest it is to modify communications and potentially provide a malicious payload to the forward units. Not only can the adversary detect locations through RF but can intercept, drop, replay, and manipulate transmissions. The protocol(s) must protect against adversarial capabilities. We can no longer rely solely on information supremacy.

1.1 Prior Research

Updating software is not a new concept, neither in the military nor in the civilian world. However, based on the new constraints and warfare environment defined above, the USMC cannot rely on the previous means it used to update software:

- In the EABO force design, units will be unable to apply updates in person with higher units.
- While past conflicts offered more time flexibility to installing updates due to the adversary's lack of technical capability, the new construct demands the ability to combat a peer adversary with similar technological capabilities. Therefore, it is pertinent to install updates as soon as possible.
- Finally, in contrast to normal software updating in commercial sectors, any radio frequency (RF) footprint generation due to bi-directional traffic poses and immediate and significant risk to operating units. Thus standard paradigms based on bi-directional client-server communications are no longer feasible.

Part of the problem is the fact that the operational systems used are designed to last years, if not decades. As this gear is going to be put up against adversaries with the ability to conduct cyber-attacks, there are new security concerns that must be corrected with updates.

Military equipment is not the only area experiencing this problem. Industrial automation and control systems are experiencing a similar threat, with adversarial attack capabilities increasing in effectiveness against older equipment. This concern is described in [2], where the authors state that software updates "have generally addressed stability and functionality rather than security". The work presents a process for installing patches and updates which includes significant testing of the update and distribution through a standardized structure in accordance with a defined policy. The proposed system is focused on policy and management but is designed for a situation with few unknown variables. The number of machines is known, the location of all machines is known and static, the activity of machines should be the same, and the RF signature is not of concern.

Conversely, this work addresses a scenario which is dynamic with many unknowns, and the back-end may not have knowledge of the state of the update recipients. For example, the number and location of units may vary, the activities of units may vary, and – crucially – the RF signature is a concern. This precludes the proposed industrial solution as a solution for the considered context.

Previous methods and policies of updating military equipment are discussed in [3]. This offers a "comprehensive mechanism for security and reliability in software deployment" which ensures that no information about the installation procedure can be learned (to protect against learning about intended hardware for military applications), and also prevents malicious users from forcing to install a package without its requirements being filled [3]. The authors' approach focuses on confidentiality and authenticity, ensuring that the installation process is not view-able and that only required packages get installed. Our reduced RF constraints, however, mean that the proposed method does not satisfy the requirements for the environment setting.

Prior work [4] addresses the issue of providing authenticity to updates to address the issue of supply chain attacks. That work proposes a system called *Dit*, a wrapper around the popular *Git* tool which adds threshold signatures which enable auditing and verification of the code throughout the development process, ensuring authenticity. In that approach, the focus is on establishing a trusted infrastructure for decentralized package managers. Additionally, the approach operates over standard HTTP protocol. In contrast, we are concerned with manipulation of the update while in transit, and the HTTP construction does not fit within the RF signature mitigation constraints.

Research has considered methods of securely updating firmware on remote embedded devices (EDs) [5], in an approach that is specifically designed for EDs that operate without an operating system. That approach relies on cryptographic primitives that run directly on the hardware to enable the EDs to provide confidentiality and verify the update is unaltered from a trusted source, and works by establishing a session key with the update provider, receiving the update, and then verifying it on hardware. The key exchange process used in [5] establishes a session key with the back-end update provider, which requires an expensive RF signature from the forward device and prevents a one-to-many broadcast from the back-end. This, does not fit the RF-minimization environment due to interaction.

A peer-to-peer (P2P) approach to file sharing in an Internet-of-Things network is detailed in [6], termed the InterPlanetary File System (IPFS). IPFS provides a means to share information among dispersed nodes. It relies on P2P sharing where the nodes comprise a Merkle DAG object - an object based on a Merkle tree which is used to account for generic data structures. IPFS also provides for versioning of its files and routing information to enable web-like sharing. However, some of its features such as version control and Merkle DAG objects introduce expensive consensus communication that is not necessary for our situation as the back-end is completely trusted. One feature included in IPFS that could be added in future work for the scenario considered in this paper is implementation of a routing table to increase efficiency when forward units are not fully connected.

1.2 Contributions

In this work, we propose a new software update protocol, comprised of three sub-protocols, that securely delivers an update to a group of participants while minimizing the RF signature of the participants. Specifically, the main contributions are:

- Back-haul Authenticated Control (BAC) Protocol: A software update protocol providing authenticated messages from a back-end to receiver participants as well as a push-button option for selective confirmation response on the received message.

- Constrained Communication Synchronization-Request (COCO-SYNC-R) Protocol: A packet sharing protocol for distribution and confirmation of receipt of the update amongst forward participants. This protocol utilizes a <u>pull</u> method for sharing packets.

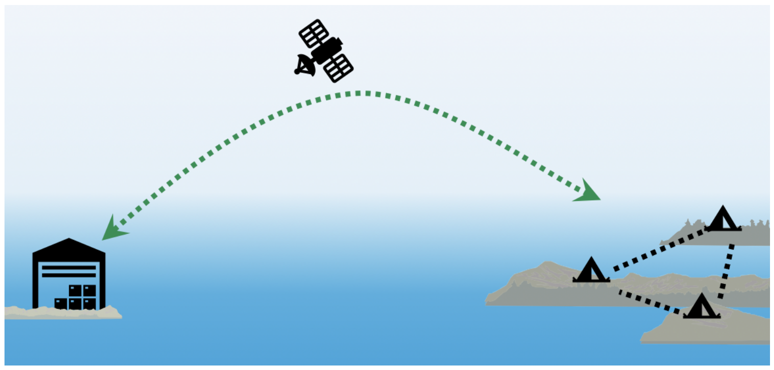


Figure 1: BAC protocol (in green) covers initial transmission of data, while the COCO-SYNC-R or COCO-SYNC-P protocols (in black) cover inter-unit distribution.

- Constrained Communication Synchronization-Provide (COCO-SYNC-P) Protocol: A packet sharing protocol for distribution and confirmation of receipt of the update amongst forward participants. This protocol utilizes a push method where one unit broadcasts to all other units.
- Security Analysis: We model the security goals of of the above protocols and prove security through computational analysis.
- RF Signature Analysis: We provide mathematical bounds for the RF signature footprint on the above protocols.

2 Protocols

Here we will describe the proposed protocols. There are three protocols outlined in this paper. The first, called Back-haul Authenticated Control (BAC), governs the authenticated data transfer from the back-end to the forward units. The other two protocols are alternatives that govern inter-unit communication for data sharing among the units. First we described the Constrained Communication Synchronization-Request (COCO-SYNC-R) protocol, which utilizes a "pull" mechanism. Second, we provide a "push" mechanism alternative protocol, Constrained Communication Synchronization-Provide (COCO-SYNC-P). These protocol roles are illustrated in Figure 1.

Since we are primarily concerned with authentication from the back-end for the update, we rely on the use of asymmetric digital signatures. We do not provide for encryption within the protocol, but it should be noted that in practice, military units use equipment with built in encryption methods [7]. The equipment exampled uses AES-256 bit symmetric encryption and relies on pre-shared Transmission Encryption Keys (TEK) [7]. This encryption method provides protection from adversaries listening in (confidentiality), but it does not provide a means to verify the identity of a sender (authenticity) nor uniqueness of the transmission (replay protection). Symmetric encryption requires additional computational resources from the equipment, but it does not significantly impact the RF signature from the forward units. In the worst case, the encryption will require an additional 255-bits per transmission. Assuming a Max Transmission Unit of 1500 Bytes (12,000 bits), the encryption will only incur a 2% increase in RF.

In the protocols, flows that contain a digital signature do not include an identity certificate from the sender. The protocol assumes that the receiver will have a list of public keys, including the back-end and forward units, and will perform an exhaustive search on this list until it finds a key that verifies the tag (with the exception of confirmation messages which attest to the sender). While this is an additional computational cost, it allows us to reduce the RF bandwidth, so the added computation is acceptable. Additionally, the public key list maintained by the forward units will be relatively small (<1000) so the added time to compute in practice will be negligible. Note, however, that this search can be avoided by the protocol implementors by inclusion an the sender identity, if desired.

Variables

- BE: Back-end Sender
- U_v: Receiver
- k: The number of forward units participating in the protocol
- N_v: A fresh nonce sampled by U_v, where $N_v \in \{1, 0\}^*$
- $Update_x$: Numeric identifier for x-th update. This may also be interpreted as Seq_Update. We abuse notation and simply use $Update$, with the sequencing of further updates handled by the application.
- M: Complete update message for the transmission

- M_n: The n-th packet of the update message
- Seq_n: Numeric sequence number for the n-th packet.
  This may also be interpreted as Seq_Packet_n.
- Seq_{total}: Total number of packets included in transmission

Throughout this paper, the term *packet* refers to a section of the overall Update message M from the back-end, i.e., the message is divided into packets M_n for transmission, where M_n corresponds to a given Seq_n and $M_1||M_2||\dots||M_n||\dots = M$ represents the full Update transmission.

2.1 BAC Protocol

Notation

$Sign_{sk_{\mathsf{BE}}}(data)$: the tag generated from signing data with BE's secret key
$\mathsf{BE} \rightarrow \mathrm{U}_v$: sending data from BE to U_v

Protocol Description

The overarching flow of the BAC protocol consists of the back-end server sending an "announcement message" followed by the update message itself in iterative packets (each individually signed), and finally a signature over the complete Update message. Ideally, the back-end would provide the entire sender side of the protocol (Flows 1-3) on repeat to reduce the number of packets missing from the receiving units' transcripts. This is due to the use environment, where dropped packets are likely with a transmission of the typical size; however, protocol repetition may not always be feasible given bandwidth constraints and is left as an application decision.

The protocol iteration completes when the back-end (BE) receives a signed confirmation message from one of the units. The confirmation message is a compilation of *requests* from each of the forward units, indicating what packets need to be re-transmitted.[1] The confirmation message is then signed by one of the forward units and sent to the back-end. Although the message is signed and sent by only one unit, it is representative of all the receiver participants, so while the transmissions $\mathsf{BE} \rightarrow \mathrm{U}_v$ are broadcast from the back-end to all forward participants, the transmission $\mathsf{BE} \leftarrow \mathrm{U}_v$ is from one unit to the back-end.

[1] If the confirmation message indicates missing packets, the implementation may select to restart the protocol sending only those packets, e.g., re-transmission of missing packets.

We present the BAC protocol below in algorithmic form, with a visualization shown in Figure 3.

Flow	**Data**
$\mathsf{BE} \rightarrow \mathrm{U}_v$	Update, Seq_{total}, $\mathrm{Sign}_{sk_{\mathsf{BE}}}$(Update, Seq_{total})
$\mathsf{BE} \rightarrow \mathrm{U}_v$	Update, Seq_n, Seq_{total}, M_n, $\mathrm{Sign}_{sk_{\mathsf{BE}}}$(Update, Seq_n, Seq_{total}, M_n)
.	$[\forall M_n \in M]$
$\mathsf{BE} \rightarrow \mathrm{U}_v$	$\mathrm{Sign}_{sk_{\mathsf{BE}}}$(Update, M)
$\mathsf{BE} \leftarrow \mathrm{U}_v$	$\mathrm{U}_v, \mathrm{N}_v$, Update, Confirmation Message*, $\mathrm{Sign}_{sk_v}(\mathrm{U}_v, \mathrm{N}_v$,
.	Update, Confirmation Message*)

*Confirmation Message:

Units	Messages
U_1	U_1, N_1, Update, $\{\mathrm{Seq}_t\}_1^*$, $\mathrm{Sign}_{sk_1}(\mathrm{U}_1$, N_1, Update, $\{\mathrm{Seq}_t\}_1^*)$
U_2	U_2, N_2, Update, $\{\mathrm{Seq}_t\}_2^*$, $\mathrm{Sign}_{sk_2}(\mathrm{U}_2$, N_2, Update, $\{\mathrm{Seq}_t\}_2^*)$
U_3	U_3, N_3, Update, $\{\mathrm{Seq}_t\}_3^*$, $\mathrm{Sign}_{sk_3}(\mathrm{U}_3$, N_3, Update, $\{\mathrm{Seq}_t\}_3^*)$
.	.
.	.
.	.
U_k	U_k, N_k, Update, $\{\mathrm{Seq}_t\}_k^*$, $\mathrm{Sign}_{sk_k}(\mathrm{U}_k$, N_k, Update, $\{\mathrm{Seq}_t\}_k^*)$

Figure 2: Confirmation Message Format

The protocol flows denoted in blue indicate the *core BAC protocol*. This protocol denotes agreement on possession of Update data packets and intent to send. The protocol on the receiver side, U_v, 'accepts' as a correct run of the protocol if at least one of the messages in blue is received. The third flow, $\mathrm{Sign}_{sk_{\mathsf{BE}}}$(Update, M), is a confirmation message on the sent Update message, for use later by receiver units, e.g., in COCO-SYNC-P or COCO-SYNC-R, as a verification to hold in memory over the entire Update message vice individual packets and can therefore be transferred and validated locally as needed. However, this last message from the back-end is not a *protocol* confirmation message, and is therefore treated as channel data. Note also, that unless all other flows from BE are received, the flow $\mathrm{Sign}_{sk_{\mathsf{BE}}}$(Update, M) cannot be validated; since we intentionally support packet dropping and use of confirmation message to request missing update packets, this third flow is not considered part of the core protocol.

Remark 2.1. *There is a space trade-off for including Seq_{total} in each mes-*

sage instead of only in the first message. For a more robust protocol, it should be included as it enables the forward units to verify the completeness of a received transmission even if the first and last packets are dropped. However, if the added space requirement is too much of a constraint, the Seq_{total} can be omitted from data messages and only kept in the header message. In this scenario, units can verify the completeness of the data if they receive either the header or the final message; in absence of both and without the inclusion of Seq_{total}, the receiver must await a re-transmission before verifying completeness.

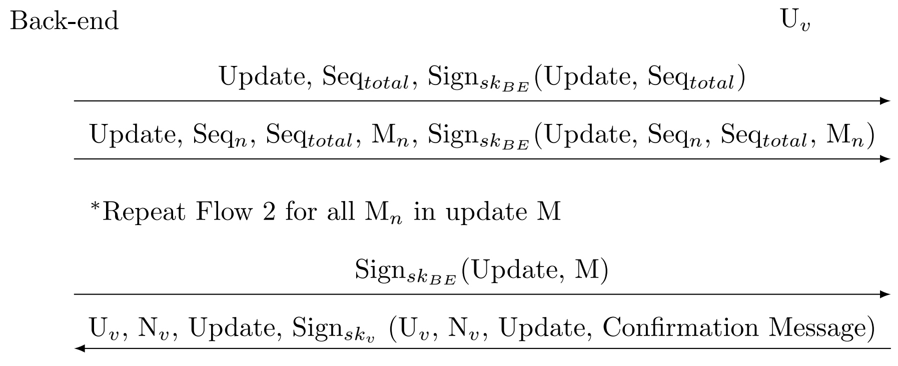

Back-end
U_v
Update, Seq_{total}, $\mathrm{Sign}_{sk_{BE}}$(Update, Seq_{total})
Update, Seq_n, Seq_{total}, M_n, $\mathrm{Sign}_{sk_{BE}}$(Update, Seq_n, Seq_{total}, M_n)
*Repeat Flow 2 for all M_n in update M
$\mathrm{Sign}_{sk_{BE}}$(Update, M)
U_v, N_v, Update, Sign_{sk_v} (U_v, N_v, Update, Confirmation Message)

Figure 3: BAC Protocol Illustration

2.2 Cache Requirements

It is expected that the forward units will maintain a cache of the signatures ($\mathrm{Sign}_{sk_{\mathsf{BE}}}$(Update, M)) received in the transmission. This is essential for local distribution so that receiver units may verify the authenticity of the entire update message once received.

Furthermore, they may hold sequence numbers, packet numbers, and packets in memory for the duration of the COCO-SYNC protocol (Seq_n, Seq_{total}, M_n). Note, however, that due to optimization of bandwidth and transmission reliability, units may re-package the update into smaller packets. While allowing flexibility, this may increase RF footprint locally, since receivers may only verify the update using $\mathrm{Sign}_{sk_{\mathsf{BE}}}$(Update, M) once the entire update is received. In the following COCO-SYNC protocol descriptions, we aim to minimize protocol flows and therefore RF footprint by leveraging the assumption that sequence numbers, packet numbers, and packets are held in memory for local distribution during the COCO-SYNC protocol, i.e., Seq_n, Seq_{total}, M_n.

2.3 COCO-SYNC-R

In both of the COCO-SYNC protocols, we present the protocol as an algorithm first, followed by a simplified flow diagram as illustration. We do this to demonstrate how w is iterated in the hierarchy throughout the protocol and explicitly show how variables are updated during the protocol. The flow diagram simplifies the protocol representation to show only the protocol flows—variable assignments are not shown in the flow diagram, but we assume that all of the variables are assigned according to the algorithmic description.

Assumptions

Sign$_{sk_{BE}}$*(data)*: the original back-end signature on the received transmission.

Variables

In addition to the variables listed previously, we add the following notation:

- $\{U_v\}$: An ordered set of the forward units engaging in protocol. The v-th unit is denoted U_v.
- $\{\mathrm{Seq}_t\}_v^*$: The set of packet sequence numbers that the v-th unit is requesting (requests denoted by *). The t-th packet's sequence number is denoted Seq_t.
- $\{\mathrm{Seq}_s\}_w$: The set of packet sequence numbers that unit w is responding with. The s-th packet's sequence number is denoted Seq_s.

We require that $\{\mathrm{Seq}_s\}_w \subseteq \{\mathrm{Seq}_t\}_v^*$. Namely, the set of sequence numbers that the w-th unit answers a request with must be a subset of the request.

Protocol Description

The following description assumes there is an execution hierarchy ordering among the units which is known by all units. Unit v is selected from the top of the hierarchy and then removed from the hierarchy for the duration of its turn. For v's turn, unit w begins at the new top of the hierarchy and is iterated through all units in the ordering (excluding v). After v completes its turn, it is added back into the hierarchy ordering at the bottom and a new target unit is selected from the top of the ordering. This approaches enforces a queued hierarchy.

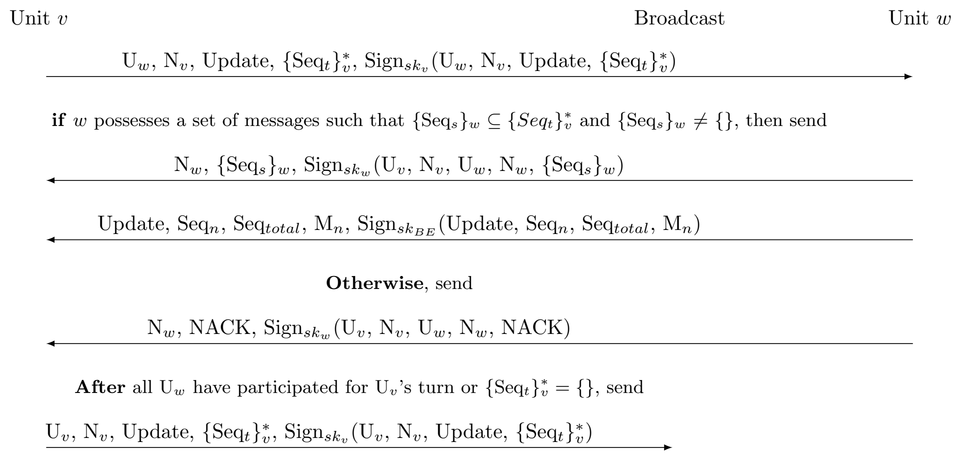
Protocol COCO-SYNC-R is described below, with an illustration provided in Figure 4.

```
Flow                          Algorithm
.                             $w \leftarrow 1$
.                             $\mathrm{U}_v$ sets $\{\mathrm{Seq}_t\}_v^*$

.                             while $\{\mathrm{Seq}_t\}_v^* \neq \{\} \wedge (w < |\{\mathrm{U}\}|)$:
$\mathrm{U}_v$                  $\mathrm{N}_v \leftarrow_\$ \{0,1\}^\lambda$
$\mathrm{U}_v \rightarrow \mathrm{U}_w$          $\mathrm{U}_w$, $\mathrm{N}_v$, Update, $\{\mathrm{Seq}_t\}_v^*$, $\mathrm{Sign}_{sk_v}(\mathrm{U}_w, \mathrm{N}_v, \mathrm{Update}, \{\mathrm{Seq}_t\}_v^*)$

.                               if $\mathrm{U}_w$ possesses $\{\mathrm{Seq}_s\}_w$ such that $\{\mathrm{Seq}_s\}_w \subseteq \{\mathrm{Seq}_t\}_v^*$
.                               and $\{\mathrm{Seq}_s\}_w \neq \{\}$:
$\mathrm{U}_w$                      $\mathrm{N}_w \leftarrow_\$ \{0,1\}^\lambda$
$\mathrm{U}_v \leftarrow \mathrm{U}_w$          $\mathrm{N}_w$, $\{\mathrm{Seq}_s\}_w$, $\mathrm{Sign}_{sk_w}(\mathrm{U}_v, \mathrm{N}_v, \mathrm{U}_w, \mathrm{N}_w, \{\mathrm{Seq}_s\}_w)$
$\mathrm{U}_v \leftarrow \mathrm{U}_w$          Update, $\mathrm{Seq}_n$, $\mathrm{Seq}_{total}$, $\mathrm{M}_n$,
.                                       $\mathrm{Sign}_{sk_{\mathsf{BE}}}(\mathrm{Update}, \mathrm{Seq}_n, \mathrm{Seq}_{total}, \mathrm{M}_n)$
.                                   [$\forall \mathrm{Seq}_n \in \{\mathrm{Seq}_s\}_w$ forwarded from $\mathrm{U}_w$'s cache of the
.                                   original back-end ($\mathsf{BE}$) broadcast.]

$\mathrm{U}_v$                      $\{\mathrm{Seq}_t\}_v^* \leftarrow \{\mathrm{Seq}_t\}_v^* \setminus \{\mathrm{Seq}_s\}_w$
.                                   [Computed updated set at $U_v$.]

.                               else
$\mathrm{U}_w$                      $\mathrm{N}_w \leftarrow_\$ \{0,1\}^\lambda$
$\mathrm{U}_v \leftarrow \mathrm{U}_w$          $\mathrm{N}_w$, NACK, $\mathrm{Sign}_{sk_w}(\mathrm{U}_v, \mathrm{N}_v, \mathrm{U}_w, \mathrm{N}_w, \mathrm{NACK})$
.                               end if
.                               $w \leftarrow w + 1$
.                             end while

$\mathrm{U}_v$                  $\mathrm{N}_v \leftarrow_\$ \{0,1\}^\lambda$
$\mathrm{U}_v \rightarrow$ broadcast  $\mathrm{U}_v$, $\mathrm{N}_v$, Update, $\{\mathrm{Seq}_t\}_v^*$, $\mathrm{Sign}_{sk_v}(\mathrm{U}_v, \mathrm{N}_v, \mathrm{Update}, \{\mathrm{Seq}_t\}_v^*)$

.                             Repeat $\forall v \in |\{U\}|$
```

As with the BAC protocol, the protocol flows denoted in blue indicate the *core COCO-SYNC-R protocol*. This protocol denotes agreement on possession of Update data packets and intent to send. We do not include the actual messages of the Update within the core protocol, as those may be dropped (leading to a break in protocol transcript matching, as described in Section 4). Also, we do not include the final broadcast message; the final broadcast message is actually collected for return as a confirmation message component in the BAC protocol, rather than a critical aspect of the COCO-SYNC-R protocol. Furthermore, neither the BAC nor the COCO-SYNC-R protocol should not fail if this message is lost due to packet dropping. If U_v's confirmation message is lost, there is no guarantee to the back-end that U_v has the Update, but if other units possess the full Update it suffices that U_v may obtain it from them. Otherwise, the lack of a full Update possessed by any unit in the BAC confirmation message indicates to the back-end possible need to re-transmit.

In addition to the above core protocol description, the implementation should include a fail-safe timeout. A timeout is crucial for added robustness to the protocol by ensuring the protocol does not hang while waiting for a unit that loses connectivity or gets jammed. The timeout should be maintained by U_v, and if no response from the current U_w is received within the timeout window, then a next U_w is selected and U_v sends its request. It would be beneficial, however, for U_v to continue listening for a late response from a previous U_w so that RF broadcasts (if answered later) are not wasted just because of the timeout.

Although the flows indicate intended partners, the data will be sent in broadcast form, so all units will be able to listen for any transmissions containing missing sequence numbers even if that unit is not acting as v. The goal of this is to reduce the number of duplicate requests. This idea is further explained in section 3.



U_w, N_v, Update, $\{\text{Seq}_t\}_v^*$, $\text{Sign}_{sk_v}(U_w, N_v, \text{Update}, \{\text{Seq}_t\}_v^*)$

if w possesses a set of messages such that $\{\text{Seq}_s\}_w \subseteq \{Seq_t\}_v^*$ and $\{\text{Seq}_s\}_w \neq \{\}$, then send

N_w, $\{\text{Seq}_s\}_w$, $\text{Sign}_{sk_w}(U_v, N_v, U_w, N_w, \{\text{Seq}_s\}_w)$

Update, Seq_n, Seq_{total}, M_n, $\text{Sign}_{sk_{BE}}(\text{Update}, \text{Seq}_n, \text{Seq}_{total}, M_n)$

Otherwise, send

N_w, NACK, $\text{Sign}_{sk_w}(U_v, N_v, U_w, N_w, \text{NACK})$

After all U_w have participated for U_v's turn or $\{\text{Seq}_t\}_v^* = \{\}$, send

U_v, N_v, Update, $\{\text{Seq}_t\}_v^*$, $\text{Sign}_{sk_v}(U_v, N_v, \text{Update}, \{\text{Seq}_t\}_v^*)$

Figure 4: COCO-SYNC-R Protocol Illustration

2.4 COCO-SYNC-P

Assumptions and Variables

We re-apply the same assumptions and variables as used in Section 2.3.

Protocol Description

The following protocol reverses the approach of COCO-SYNC-R. Instead of having each unit request what it needs from every other unit, all units will broadcast what they are missing at the beginning. Then, the unit with the most complete transcript will be selected as the lead unit and broadcast the missing sequence numbers it has to the other units. The protocol is designed to be iterative, and the protocol can be repeated until matching transcripts are achieved among all units, or an exit condition is set. More on this in remark 2.2. Color coding in blue for the *core COCO-SYNC-P protocol* follows that of BAC and COCO-SYNC-R protocol.

Flow	**Algorithm**
U_w	$\mathrm{N}_w, \leftarrow_{\$} \{0,1\}^\lambda$
$\mathrm{U}_w \rightarrow$ broadcast	N_w, Update, $\mathrm{Sign}_{sk_w}(\mathrm{U}_w, \mathrm{N}_w, \text{Update})$
.	**For all $\mathbf{U}_u \in \{\mathbf{U}\}$:**
U_u	$\mathrm{N}_u \leftarrow_{\$} \{0,1\}^\lambda$
$\mathrm{U}_u \rightarrow$ broadcast	U_u, N_u, Update,
.	$\{\mathrm{Seq}_s\}^*_u$, $\mathrm{Sign}_{sk_u}(\mathrm{U}_u, \mathrm{N}_u, \text{Update}, \{\mathrm{Seq}_s\}^*_u)$
.	**End for**
.	$(U_v, t) \leftarrow (U_u, s).[\min_{\forall \mathrm{U}_u \in \{\mathrm{U}\}}\{\lvert\{\text{Update.Seq}_s\}^*_u\rvert\} \wedge \mathsf{cond}_{\mathsf{Lead}}]$
.	$\mathsf{Lead} \leftarrow (U_v, t)$
.	$\{\mathrm{Seq}_t\}_v \leftarrow \left\{ \bigcup_{u=1}^{\lvert\{U\}\rvert} \{\mathrm{Seq}_s\}^*_u \right\} \setminus \{\mathrm{Seq}_t\}^*_v$
U_v	$\mathrm{N}_v \leftarrow_{\$} \{0,1\}^\lambda$
$\mathrm{U}_v \rightarrow$ broadcast	N_v, N_w, $\{\mathrm{Seq}_t\}_v$, $\mathrm{Sign}_{sk_v}(\mathrm{U}_v, \mathrm{N}_v, \mathrm{U}_w, \mathrm{N}_w, \{\mathrm{Seq}_t\}_v)$
.	**If** $\{\mathrm{Seq}_t\}_v == \{\} \vee \mathsf{cond}_{\mathsf{exit}}$:
.	**exit**
$\mathrm{U}_v \rightarrow$ broadcast	Update, Seq_n, Seq_{total},
.	M_n, $\mathrm{Sign}_{sk_{\mathsf{BE}}}(\text{Update}, \mathrm{Seq}_n, \mathrm{Seq}_{total}, \mathrm{M}_n)$
.	$[\forall \mathrm{Seq}_n \in \{\mathrm{Seq}_t\}_v]$

Repeat

Remark 2.2. *The protocol can be re-run (i.e., corresponding to* **Repeat***), until all units' transcripts match or until predetermined number of iterations are reached. The former case is captured by* $\{\mathrm{Seq}_t\}_v == \{\}$ *which holds if the request sets from all units were identical (whether from obtaining the full Update, partial but identical packet sets, or no update packets). In the worse case, units will have received the BAC message indicating that an Update would be set, but non of the ensuing Update packets.*

Protocol exit after a certain number of iterations can be enforced via $\mathsf{cond}_{\mathsf{exit}}$*.*

Remark 2.3. *We allow a tie-breaker condition* $\mathsf{cond}_{\mathsf{Lead}}$*, such that if more than one unit has an equal number of minimum requested packets. Alternatively, if* (U_u, s) *has already been the* Lead*, the condition can force a re-calibration of hierarchy. For simplicity, we let* $\mathsf{cond}_{\mathsf{Lead}} = 1$ *for the remainder of this paper, but leave it up to the implementation for appropriate conditions to handle other such cases.*

If packet drops occur when units are broadcasting their request sets, it is possible that more than one unit may claim the role as Lead. To account for this, a timeout should be enforced following Flow 3 where a unit broadcasts intent to be Lead. The timeout will allow all units that may have claimed lead to examine the other broadcast commit sets and cede the role of Lead the the unit with the most complete commit set. In the instance of more than one unit having commit sets of the same size, the role of Lead will go to the unit highest in the hierarchy.

One benefit to this protocol is that it only highlights the RF signature of the lead unit. So if one unit is in a position where it is safer to broadcast RF, that unit could always be selected as the lead (U_v). Additionally, the protocol can be adapted to add a step where the lead unit requests missing sequence numbers from units it knows has them based on the initial request broadcasts. That way, the lead unit has the most complete transcript possible before it broadcasts.

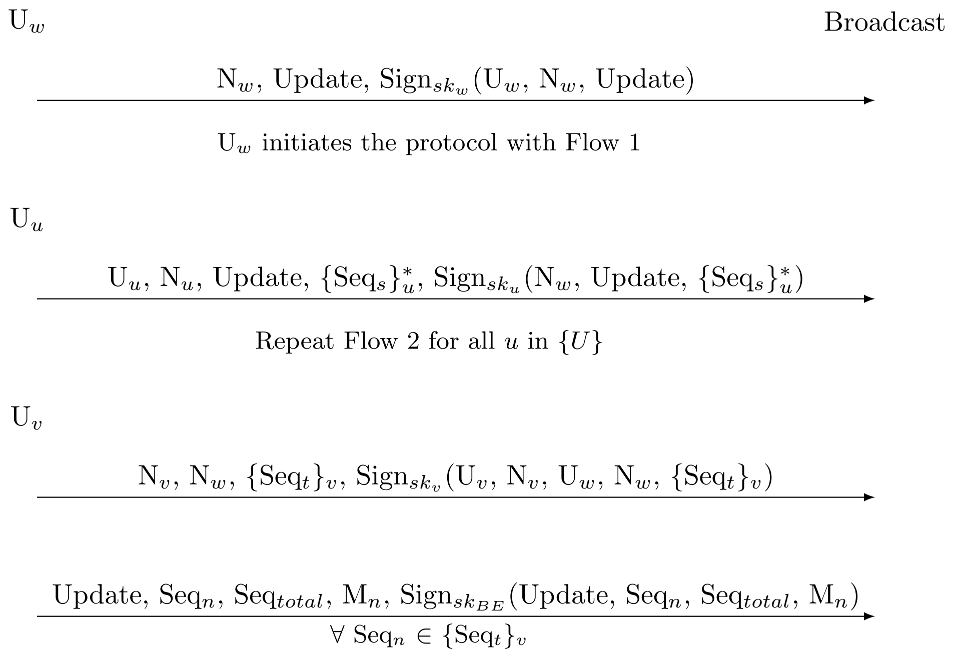

U_w
Broadcast
N_w, Update, $\text{Sign}_{sk_w}(U_w, N_w, \text{Update})$
U_w initiates the protocol with Flow 1
U_u
U_u, N_u, Update, $\{\text{Seq}_s\}^*_u$, $\text{Sign}_{sk_u}(N_w, \text{Update}, \{\text{Seq}_s\}^*_u)$
Repeat Flow 2 for all u in $\{U\}$
U_v
N_v, N_w, $\{\text{Seq}_t\}_v$, $\text{Sign}_{sk_v}(U_v, N_v, U_w, N_w, \{\text{Seq}_t\}_v)$
Update, Seq_n, Seq_{total}, M_n, $\text{Sign}_{sk_{BE}}(\text{Update}, \text{Seq}_n, \text{Seq}_{total}, M_n)$
$\forall\ \text{Seq}_n \in \{\text{Seq}_t\}_v$

Figure 5: COCO-SYNC-P Protocol

3 Examples

The issue of inter-unit communication represents a challenge when addressing this scenario. The forward units in question are likely experiencing dynamic situations that cannot be fit into definitive solutions (e.g. via predictable connection among all forward parties). The units are further limited by the necessity to maintain low RF signatures, communication equipment that may be compromised or generally inferior, and terrain not conducive to communication. Due to this complexity and range of situations, it is misguided to assume one protocol will be optimal for all scenarios. Units must be able to adapt to the environment and communicate amongst themselves. Thus, we have created two separate protocols that capture two distinct methods for inter-unit communications. Selection among these may be made based on assessed optimality for a given scenario. These two methods, presented in Section 2.3 and Section 2.4, are visualized below with examples. It is important to emphasize that these are not technical depictions of the protocol, but rather high level scenario depictions to illustrate the overarching concepts.

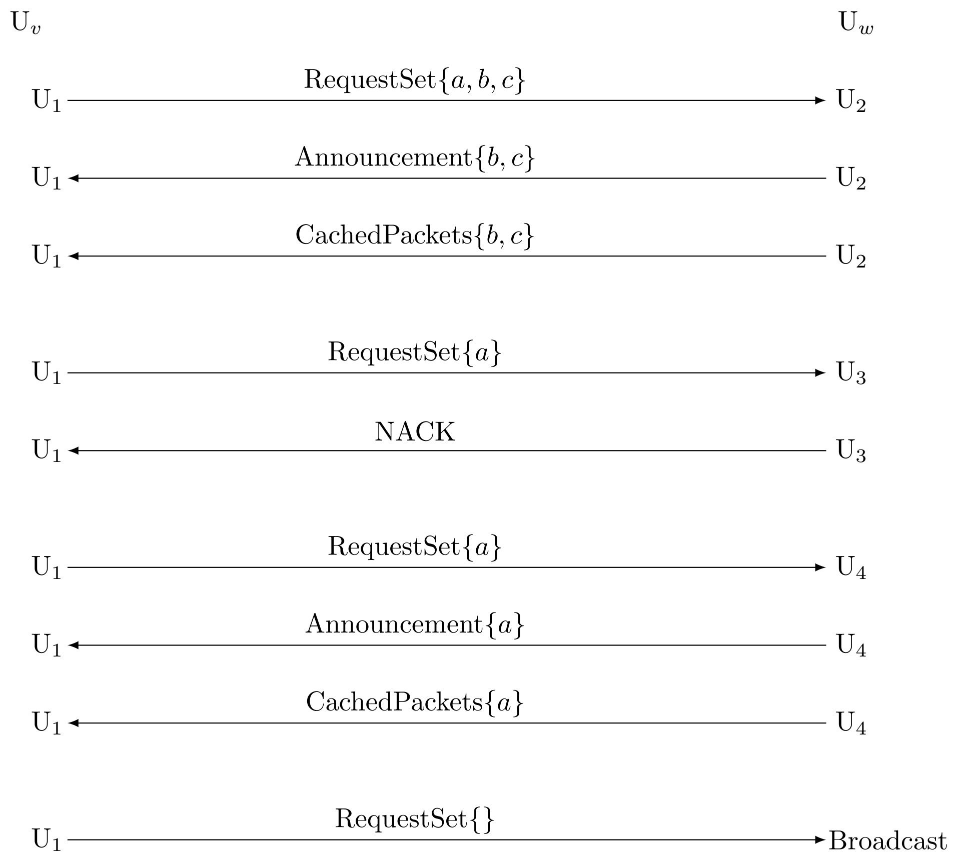
At a basic level, the COCO-SYNC-R protocol follows a pull mechanism and relies on a pre-established hierarchy of units. Figure 6 shows an example scenario. Assume there are five forward units, depicted as Units 1-5, which need to affirm receipt of a shared update. The update itself consists of packets *a*-*e*. For this example, we will assume Unit 1 is the unit exercising its 'turn', followed by Unit 2, and so on (i.e. 1-5 is the listed hierarchy). For both examples, we will say that units successfully received the following packets:

- Unit 1: $\{d, e\}$
- Unit 2: $\{b, c, d, e\}$
- Unit 3: $\{c, d, e\}$
- Unit 4: $\{a, b, d\}$
- Unit 5: $\{b, c, d\}$

After receiving the final flow of the BAC protocol consisting of the update from headquarters, Unit 1 sends labels (packet numbers) to Unit 2 for the packets it is missing. In this case, Unit 1 will send Unit 2 that it is missing $\{a, b, c\}$. Unit 2 then replies with the subset list of labels $\{b, c\}$ in its cache, followed by original packets as sent by the back-end.

At this point, any unit missing b or c that sees Unit 2's message will receive the packets as well.[2] So Unit 3 gets b and Unit 4 gets c. Unit 1, which is still in need of packet a, will then send Unit 3 its new request set (which is just the label for a). At this point, Unit 3 responds with its subset list of labels followed by the corresponding packets in its cache. If a unit does not have any of the requested packets, the unit will respond with a NACK. This process will continue until Unit 1 has nothing left to request, requesting the empty set, or Units 2-5 have responded. The process will then start over with Unit 2 as the lead unit, requesting all missing packets from other units according to the pre-established ordering. The entire protocol can then be repeated until all units request the same request set, or for a predetermined number of iterations.

[2]We call this an *observing case* for which we will consider RF footprint as distinct from a *non-observing case* in Section 5.4.


U_v
U_w
U_1 RequestSet$\{a, b, c\}$ U_2
U_1 Announcement$\{b, c\}$ U_2
U_1 CachedPackets$\{b, c\}$ U_2
U_1 RequestSet$\{a\}$ U_3
U_1 NACK U_3
U_1 RequestSet$\{a\}$ U_4
U_1 Announcement$\{a\}$ U_4
U_1 CachedPackets$\{a\}$ U_4
U_1 RequestSet$\{\}$ Broadcast

At this point, Unit 1 has requested the empty set so its turn ends and Unit 2 begins to request

Figure 6: COCO-SYNC-R Flow Diagram for Example Case

The COCO-SYNC-P protocol mirrors COCO-SYNC-R but follows a push structure. It features an agreed-upon method to determine the lead unit and relies on that unit acting as a forward hub. After receiving the final flow of the BAC protocol, a unit within $\{U\}$ will begin the protocol with an initiation message. After the protocol is begun, all units will broadcast a request set consisting of the packets they are missing. The unit missing the least number of packets is then assigned to be the lead. The lead unit will then broadcast the packets it possesses which have been seen in other units' request sets. The protocol is designed to be iterative and can be repeated until every unit broadcasts the same request set or for a determined number of iterations.

To visualize it, we can refer to the same example, with five units, labeled 1-5, needing packets *a*-*e* where the units' transcripts are identical to those above. After receiving the final message, one of the units, for this example it will be Unit 3, sends the initiation message. Then, Unit 1 broadcasts labels (packet numbers) for packets $\{a, b, c\}$, signaling it is missing packets a, b, c; Unit 2 broadcasts label $\{a\}$; Unit 3 broadcasts labels $\{a, b\}$; Unit 4 broadcasts labels $\{c, e\}$; and Unit 5 broadcasts labels $\{a, e\}$. The units all recognize that Unit 2 will take lead because it has the most complete update (note that if multiple units had the same number of packets missing, they would refer to a predetermined order again). Unit 2 then sends packets $\{b, c, d, e\}$ as a broadcast to all other units. The protocol can then be repeated where Unit 4 is selected as lead and broadcasts labels $\{a\}$.

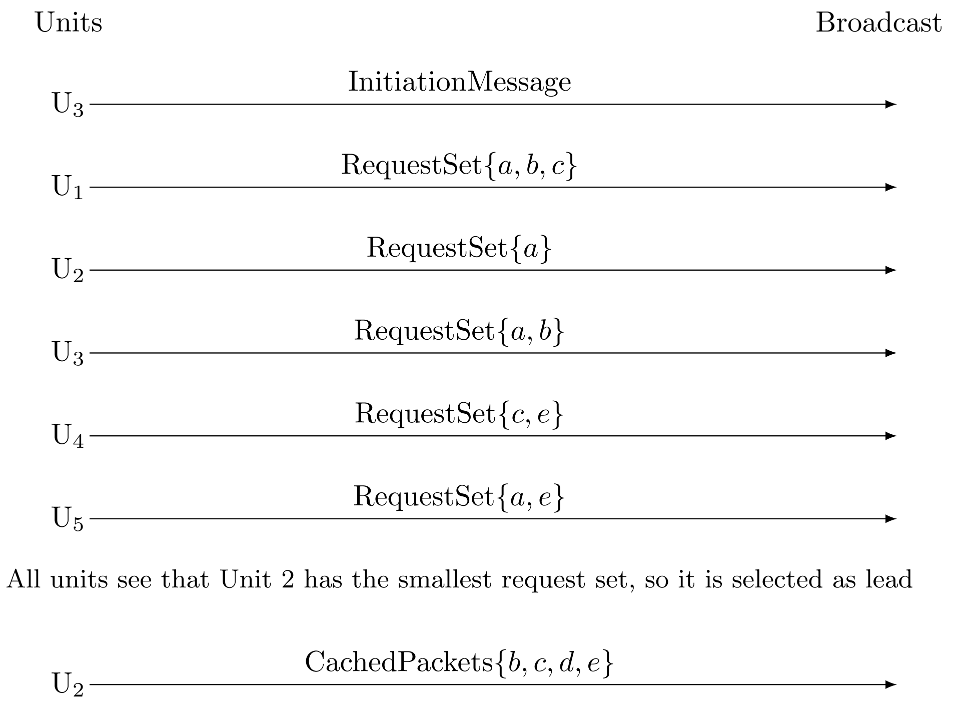


Figure 7: COCO-SYNC-P Flow Diagram for Example Case

4 Security Model

To perform an analysis on the three protocols, we will first describe the security model consisting of protocol goals and adversarial capabilities. In analyzing security, we will demonstrate the required steps and computational capabilities that an adversary must have to compromise the protocol. Finally, we will also bound the RF footprint created from these protocols.

Protocol Agnostic Variables In our model, an adversary $\mathcal{A}$ interacts with several sessions of multiple identities running the protocol. We abuse notation and denote by π_i^v the i-th session of identity U_v running the protocol. We adapt the notation of [8] and denote $\mathcal{I} = \mathcal{R} \cup \mathcal{S}$ as the set of *identities* modeling both various recipients ($\mathcal{R} = \{\mathrm{U}_i\}$) and the back-end ($\mathcal{S} = \{\mathsf{BE}\}$) in the system, each identity $\mathrm{U}_v \in \mathcal{I}$ being associated with a secret/public key pair (sk_v, pk_v). Each identity also has access to the public keys of all other identities. We associate with each session the following protocol-agnostic internal state variables:

- $\mathsf{id} \in \mathcal{I}$ indicates the owner of the session (e.g., $\mathsf{id} = \mathrm{U}_v$ for a session π_i^v).
- $\mathsf{trans} \in \{0,1\}^* \cup \{\bot\}$ records the sent and received messages.
- $\mathsf{ad} \in \{0,1\}^*$ assigns additional, authenticated data provided by the protocol participants. The content of ad is not specified within the protocol (e.g. Update message content); however, we consider ad to be a sub-component of the sent and received messages, trans.
- $\Lambda \in \{\mathtt{accept}, \mathtt{reject}\}$.

We write, e.g., $\pi_i^v.\mathsf{trans}$ when referring to state variables of a specific session.

Adversarial Capabilities The more capabilities an adversary is granted, the stronger the adversary is considered. We will assume an adversary-as-the-network paradigm, where $\mathcal{A}$ controls the network and possesses a standard set of the capabilities to delete, read, modify, and replay messages and create new session instances. Additionally, we allow specific queries to the adversary, adapting from [9]:

- $\mathsf{Corrupt}(\mathrm{U}_v)$: corrupts the long-term private key corresponding to identity $\mathrm{U}_v \in \mathcal{I}$. If $\mathsf{Corrupt}(\mathrm{U}_v)$ is the τ-th query of $\mathcal{A}$, we say that U_v is τ-corrupted. For parties that are uncorrupted, we set $\tau := \infty$.

- $\mathsf{Send}(\pi_i^v, M)$: The adversary may use a Send query to send a message M to session π_i^v. The session oracle will respond according to the protocol specification, depending on its internal state.
  - **BAC Protocol:**
    If the adversary asks Send to oracle π_i^v, the oracle first checks if M consists of an initiation symbol and Update message pair $(\top, m)$, and if so sets $\mathsf{role} = \mathsf{BE}$ and responds with the first message in the protocol based on Update m. Else it sets $\mathsf{role} = \mathsf{recipient}$ and responds according to the protocol.
  - **COCO-SYNC-R Protocol:**
    If the adversary asks Send to oracle π_i^v, the oracle first checks if M consists of an initiation symbol and Update message pair $(\top, m)$, and if so sets $\mathsf{role} = \mathsf{sender}$ and responds with the first message in the protocol based on Update m. Else it sets $\mathsf{role} = \mathsf{receiver}$ and responds according to the protocol.
  - **COCO-SYNC-P Protocol:**
    If the adversary asks Send to oracle π_j^w, the oracle first checks if M consists of an initiation symbol and Update message pair $(\top, m)$, and if so sets $\mathsf{role} = \mathsf{sender}$ and responds with the first message in the protocol based on the Update m. Else it responds according to the protocol.

    If, during the protocol run, $\mathsf{Lead} \leftarrow (U_v, t)$, then for π_i^v we set $\mathsf{role} = \mathsf{receiver}$.

For both the BAC protocol and COCO-SYNC-P protocol, the receiver is specified during the course of the protocol run, and is not immediately known to the sender.

Following [9], we denote the transcript at a session by $\pi_i^v.\mathsf{trans}$, a sequence that consists of all messages sent and received by π_i^v in chronological order (not including the initialization-symbol). We further clarify assignment to trans as plaintext messages in pre-processing (sender-side) and post-processing (receiver side) form.

We say that $\pi_i^v.\mathsf{trans}$ is a prefix of $\pi_j^w.\mathsf{trans}$, if $\pi_i^v.\mathsf{trans}$ contains at least one message and the messages in $\pi_i^v.\mathsf{trans}$ are identical to and in the same order as the first $|\pi_i^v.\mathsf{trans}|$ messages of $\pi_j^w.\mathsf{trans}$.

Definition 4.1 (Matching Conversations [9]). *We say that π_i^v has a* matching conversation *with π_j^w, if*

- $\pi_j^w.\mathsf{trans}$ *is a prefix of* $\pi_i^v.\mathsf{trans}$ *and* $\pi_i^v.\mathsf{trans}$ *sent the last message, or*
- $\pi_i^v.\mathsf{trans} = \pi_j^w.\mathsf{trans}$ *and* $\pi_j^w.\mathsf{trans}$ *sent the last message.*

Definition 4.2. *We say that a protocol* Π *between two parties* π_i^v *and* π_j^w*, run under a 'benign' adversary* $\mathcal{A}$ *that faithfully delivers messages through a series of* Send *queries, is* correct *if both oracles set* $\Lambda = \mathtt{accept}$.

BAC Security

We associate with each session the additional following internal state variables:

- $\mathsf{role} \in \{\mathsf{BE}, \mathsf{recipient}\}$ indicates the entity's role in the session. We require that $\mathsf{role} = \mathsf{recipient}$ (resp. $\mathsf{role} = \mathsf{BE}$) if and only if $\mathsf{id} \in \mathcal{R}$ (resp. $\mathsf{id} = S \in \mathcal{S}$).
- $\mathsf{pid} \in \mathcal{I} \cup \{\perp\}$ indicates the communication partner, and is set exactly once per sub-protocol run. Initially, $\mathsf{pid} = \perp$ can be set (if $\mathsf{role} = \mathsf{BE}$) to indicate that the receiver's identity is to be learned within the protocol (i.e., post-specified).

Definition 4.3 (Mutual Authentication Security (adapted from [9]))**.** *We say that an adversary* BAC*-*(t, ϵ)*-breaks a mutual authentication (MA) protocol* Π *run on security parameter* λ*, if* $\mathcal{A}$ *runs in time* t *and when* $\mathcal{A}$ *terminates, then with probability at least* ϵ *there exists an oracle* π_i^v *such that*

- π_i^v *'accepts' when* $\mathcal{A}$ *issues its* τ*-th query with partner* $\mathsf{pid} = U_w$*, and*
- U_w *is* τ_w*-corrupted with* $\tau < \tau_w$*, and*
  - *If* $\pi_i^v.\mathsf{role} = \mathsf{BE}$*: there is no oracle* π_j^w *such that* π_i^v *has a matching conversation to* π_j^w.
  - *If* $\pi_i^v.\mathsf{role} = \mathsf{recipient}$*: there is no unique oracle* π_j^w *such that* π_i^v *has a matching conversation to* π_j^w.

If an oracle π_i^v *accepts in the above sense, then we say that* π_i^v *accepts maliciously. We say that an MA protocol is* BAC*-*(t, ϵ)*-secure, if it is correct and there exists no adversary that* BAC*-*(t, ϵ)*-breaks it. We define the adversarial advantage of* $\mathcal{A}$*,* $Adv_{\Pi,\mathcal{A}}^{\mathsf{BAC}}$*, as the probability that* $\mathcal{A}$ BAC*-*(t, ϵ)*-breaks it.*

COCO-SYNC-R Security

We associate with each session the following additional internal state variables:

- $\mathsf{role} \in \{\mathsf{sender}, \mathsf{receiver}\}$ indicates the entity's role in the session.
- $\mathsf{pid} \in \mathcal{I} \cup \{\bot\}$ indicates the communication partner, and is set exactly once per sub-protocol run.

The remaining aspects of modeling for COCO-SYNC-R reflect those used in the BAC model above, with the following adaptation to Definition 4.5:

Definition 4.4 (Mutual Authentication Security (adapted from [9])). *We say that an adversary COCO-SYNC-R-(t, ϵ)-breaks an MA protocol Π run on security parameter λ, if $\mathcal{A}$ runs in time t and when $\mathcal{A}$ terminates, then with probability at least ϵ there exists an oracle π_i^v such that*

- *π_i^v 'accepts' when $\mathcal{A}$ issues its τ-th query with partner $\mathsf{pid} = U_w$, and*
- *U_w is τ_w-corrupted with $\tau < \tau_w$, and*
- *There is no unique oracle π_j^w such that π_i^v has a matching conversation to π_j^w.*

If an oracle π_i^v accepts in the above sense, then we say that π_i^v accepts maliciously. We say that an MA protocol is COCO-SYNC-R-(t, ϵ)-secure, if it is correct and there exists no adversary that COCO-SYNC-R-(t, ϵ)-breaks it. We define the adversarial advantage of $\mathcal{A}$, $Adv_{\Pi,\mathcal{A}}^{\mathsf{COCO\text{-}SYNC\text{-}R}}$, as the probability that $\mathcal{A}$ COCO-SYNC-R-(t, ϵ)-breaks it.

The above differs from MA security under the BAC model in that we require uniqueness of the partner oracle. Under BAC, any recipient may have a matching transcript (similarly how distributed units may respond with a confirmation message). Under COCO-SYNC-R, following a 'pull mechanism' assumption, we require that exactly one unit responds.

COCO-SYNC-P Security

We associate with each session the following additional internal state variables:

- $\mathsf{role} \in \{\mathsf{sender}, \mathsf{receiver}\}$ indicates the entity's role in the session.

- $\mathsf{pid} \in \mathcal{I} \cup \{\perp\}$ indicates the communication partner, and is set exactly once per sub-protocol run. Initially, $\mathsf{pid} = \perp$ can be set (if $\mathsf{role} = \mathsf{sender}$) to indicate that the receiver's identity is to be learned within the protocol (i.e., post-specified).

The remaining aspects of modeling for COCO-SYNC-P reflect those used in the BAC model above, with the following adaptation to Definition 4.5, namely removing the requirement for a unique partner oracle if the session is in a sender role.[3]

Definition 4.5 (Mutual Authentication Security (adapted from [9])). *We say that an adversary COCO-SYNC-P-(t, ϵ)-breaks an MA protocol Π run on security parameter λ, if $\mathcal{A}$ runs in time t and when $\mathcal{A}$ terminates, then with probability at least ϵ there exists an oracle π_i^v such that*

- *π_i^v 'accepts' when $\mathcal{A}$ issues its τ-th query with partner $\mathsf{pid} = U_w$, and*
- *U_w is τ_w-corrupted with $\tau < \tau_w$, and*
  - *If $\pi_i^v.\mathsf{role} = \mathsf{sender}$: there is no oracle π_j^w such that π_i^v has a matching conversation to π_j^w.*
  - *If $\pi_i^v.\mathsf{role} = \mathsf{receiver}$: there is no unique oracle π_j^w such that π_i^v has a matching conversation to π_j^w.*

If an oracle π_i^v accepts in the above sense, then we say that π_i^v accepts maliciously. We say that an MA protocol, Π is COCO-SYNC-P-(t, ϵ)-secure, if it is correct and there exists no adversary that COCO-SYNC-P-(t, ϵ)-breaks it. We define the adversarial advantage of $\mathcal{A}$, $Adv_{\Pi,\mathcal{A}}^{\mathsf{COCO\text{-}SYNC\text{-}P}}$, as the probability that $\mathcal{A}$ COCO-SYNC-P-(t, ϵ)-breaks it.

5 Analysis

In this section we analyze the three protocols under the respective models described in Section 4. The security of packet transmission in the channel correlates to the EUF-CMA security of the digital signature (see Appendix 7.1 for definition). This section's analyses correspond to the protocol security of the communicating parties, and the proofs follow similarly to that of [10] and [11].

[3]In COCO-SYNC-P, multiple receivers may self-identify for the receiver role due to disconnected networks, and we consider this as acceptable execution for obtaining a maximum number of missing packets, accepting that the format of COCO-SYNC-P already comes at an added RF cost over COCO-SYNC-R.

5.1 BAC Protocol Analysis

The BAC protocol focuses on authenticity of the update packets as exemplified by the BAC security model. We are not concerned about replays of these packets as replays will not cause unnecessary responses which would increase RF footprint among the receiving units, i.e. units can choose to accept or reject authenticated packets based on whether the packets are needed. In fact, the BAC protocol design is intended to specifically support transmission to cover cases of missing packets.
Freshness of the protocol and data authenticity on the associated data as well as entity authenticity is required from BE. Freshness and entity authenticity from U_v is required by BE for termination of the protocol. The transcript $\pi_i^v.\mathsf{trans}$ (resp. $\pi_j^w.\mathsf{trans}$) is defined across the BAC protocol flows shown in blue (see Section 2.1).

Theorem 5.1 (Security of BAC)**.** *Let $\mathcal{A}$ be a PPT adversary against the BAC security of the BAC protocol Π, running in time $t' \approx t$. Then*

$$Adv^{\mathsf{BAC}}_{\Pi,\mathcal{A}}(\lambda) = q^2 \cdot 2^{-\lambda} + (Seq_{total} + 2) \cdot n_{\mathcal{I}} n_s^2 \cdot Adv^{SUF-CMA}_{\mu,\mathcal{B}}(\lambda)$$

where q is a polynomial bound on the number of queries allowed to $\mathcal{A}$, Seq_{total} is the total packets that the Update message transmission is parsed into, $n_{\mathcal{I}} = |\mathcal{I}|$ is the number of identities, $n_s = |[s]|$ the maximum number of sessions at an identity, and μ is a signature scheme.

Proof. We perform analysis as an experiment as a series of games, starting with the real security experiment and introducing modifications in each game until a final bound on the adversary, $\mathcal{A}$'s, success is reached. The adversarial advantage in Game i is denoted Adv^i.

Game 0. This game is the same as the original experiment:

$$\mathrm{Adv}^0_{\mathcal{A}} = \mathrm{Adv}^{\mathsf{BAC}}_{\Pi,\mathcal{A}}(\lambda)$$

Game 1. This game proceeds as in Game 1, but we abort the experiment if any two sessions select the same nonce.

$$\mathrm{Adv}^0 \leq q^2 \cdot 2^{-\lambda} + \mathrm{Adv}^1 \quad .$$

Game 2. This game proceeds as in Game 1, except that the challenger guesses $\mathrm{U}_v \in \mathcal{I}$ where $|\mathcal{I}| = n_{\mathcal{I}}$ and sessions $i, j \in [s]$ where $|[s]| = n_s$, i.e., the party identities and sessions that $\mathcal{A}$ will try to win the BAC experiment against. We raise an `abort` clause and end the experiment if $\mathcal{A}$ does not try to win against the guessed sessions. Thus,

$$\mathrm{Adv}^1 \leq n_{\mathcal{I}} n_s^2 \cdot \mathrm{Adv}^2 \quad .$$

We now provide the final bound to the adversarial probability of winning BAC, i.e., the adversary attempts to win by getting the sender or receiver to accept maliciously. We break this game into two cases, based on sender/receiver roles:

Case 1: Receiver accepts maliciously. At some point U_v receives a flow of the form Update, ad, $\mathrm{Sign}_{sk_{\mathsf{BE}}}(\mathrm{Update},\ \mathsf{ad})$ with associated data corresponding to Seq_{total} or Seq_n, Seq_{total}, M_n. U_v may receive all messages in the first and second core BAC flows, or only one.
U_v accepts maliciously with $\mathsf{pid} = \mathsf{BE}$ if BE is uncorrupted and there is no unique oracle π_j^{BE} with matching conversation to π_i^v. This implies that the adversary would need to forge the message Update, ad, $\mathrm{Sign}_{sk_{\mathsf{BE}}}(\mathrm{Update},\ \mathsf{ad})$, such that the received message differs or that the verification holds for an identity other than BE (as identities are already addressed in Game 2). However, that in turn implies ability to create an adversary, $\mathcal{B}$ against the strong unforgeability of the signature. Thus,

$$\mathrm{Adv}_{\mathcal{A}}^{2-recv} \leq (\mathrm{Seq}_{total} + 1) \cdot \mathrm{Adv}_{\mathcal{B}}^{SUF-CMA}(\lambda) \quad .$$

Case 2: Sender accepts maliciously. At some point BE receives a flow of the form U_v, N_v, Update, ad, $\mathrm{Sign}_{sk_v}(\mathrm{U}_v, \mathrm{N}_v,\ \mathsf{ad})$, with associated data corresponding to the Confirmation Message. BE sets $\mathsf{pid} = \mathrm{U}_v$, checks the validity of the signature under the public signing key of U_v, and accepts maliciously with $\mathsf{pid} = \mathrm{U}_v$ if U_v did not send a message of this form. This in turn implies ability to create an adversary, $\mathcal{B}$ against the strong unforgeability of the signature, and thus

$$\mathrm{Adv}_{\mathcal{A}}^{2-send} \leq \mathrm{Adv}_{\mathcal{B}}^{SUF-CMA}(\lambda) \quad .$$

□

Remark 5.1. *Since we allow for a protocol run to 'accept' if any of the first two flow back-end messages are received, and these are each individually signed to support better functionality on the receiver side, the worst-case* *adversarial advantage gains a factor of Seq_{total}. However, it is worth noting that $n_{\mathcal{I}}$ is likely quite small, and the fixed* BE *identity also removed a typical reduction factor of $n_{\mathcal{I}}$. Thus, the overall reduction may be considered reasonably tight.*

5.2 COCO-SYNC-R Security Analysis

We restrict analysis of COCO-SYNC-R to the core protocol, i.e., the flows depicted in blue in the protocol description. Inter-unit transmission of the update packets is protected by the back-end's signature in that Update messages as part of the BAC protocol. We do not require replay protection, etc., on such packets as re-receipt and authenticity of the source unit relay of the Update message is not considered a security threat in our model as long as the BAC transmission itself is authentic. Thus, the COCO-SYNC-R core protocol focuses on authenticity and freshness of the request-to-send packets inter-unit, as responses to replays of such requests could cause unnecessary RF footprint. Similarly, authenticity of the receiver and associated data establishes knowledge of what packets the receiver may (resp. may not) provide. Transcript $\pi_i^v.\mathsf{trans}$ (resp. $\pi_j^w.\mathsf{trans}$) is defined across the COCO-SYNC-R protocol flows shown in blue.

Theorem 5.2 (Security of COCO-SYNC-R)**.** *Let $\mathcal{A}$ be a PPT adversary against the COCO-SYNC-R security of the COCO-SYNC-R protocol Π, running time $t' \approx t$. Then*

$$Adv_{\Pi,\mathcal{A}}^{\textsf{\textit{COCO-SYNC-R}}}(\lambda) = q^2 \cdot 2^{-\lambda} + 2n_{\mathcal{I}}^2 n_s^2 \cdot Adv_{\mu,\mathcal{B}(\lambda)}^{SUF-CMA}$$

where q is a polynomial bound on the number of queries allowed to $\mathcal{A}$, $n_{\mathcal{I}} = |\mathcal{I}|$ is the number of identities, $n_s = |[s]|$ the maximum number of sessions at an identity, and μ is a signature scheme.

Proof. We perform analysis as an experiment as a series of games, starting with the real security experiment and introducing modifications in each game until a final bound on the adversary, $\mathcal{A}$'s, success is reached. The adversarial advantage in Game i is denoted Adv^i.

Game 0. This game is the same as the original experiment:

$$\mathrm{Adv}^0 = \mathrm{Adv}_{\Pi,\mathcal{A}}^{\textsf{COCO-SYNC-R}}(\lambda) \quad .$$

Game 1. This game proceeds as in Game 1, but we abort the experiment if any two sessions select the same nonce.

$$\mathrm{Adv}^0 \leq q^2 \cdot 2^{-\lambda} + \mathrm{Adv}^1 \quad .$$

Game 2. This game proceeds as in Game 1, except that the challenger guesses $v, w \in \mathcal{I}$ where $|\mathcal{I}| = n_{\mathcal{I}}$ and sessions $i, j \in [s]$ where $|[s]| = n_s$, i.e., the party identities and sessions that $\mathcal{A}$ will try to win the COCO-SYNC-R experiment against. We raise an `abort` clause and end the experiment if $\mathcal{A}$ does not try to win against the guessed sessions. Thus,

$$\mathrm{Adv}^1 \leq n_{\mathcal{I}}^2 n_s^2 \cdot \mathrm{Adv}^2 \quad .$$

We now provide the final bound to the adversarial probability of winning COCO-SYNC-R, i.e., the adversary attempts to win by getting the sender or receiver to accept maliciously. We break this game into two cases, based on sender/receiver roles:
Case 1: Receiver accepts maliciously. At some point U_w receives a flow of the form $\mathrm{U}_w,\ \mathrm{N}_v,\ \mathsf{ad},\ \mathrm{Sign}_{sk_v}(\mathrm{U}_w,\ \mathrm{N}_v,\ \mathsf{ad})$ (with associated data corresponding to the components of $\mathrm{Update},\ \{\mathrm{Seq}_t\}^*_v$. U_w performs a search and verification check of the signature over all identities in $\mathcal{I}$. U_w accepts maliciously with $\mathsf{pid} = \mathrm{U}_v$ if U_v is uncorrupted and there is no unique oracle π_i^v with matching conversation to π_j^w. This implies that the adversary would need to forge the message $\mathrm{U}_w,\ \mathrm{N}_v,\ \mathsf{ad},\ \mathrm{Sign}_{sk_v}(\mathrm{U}_w,\ \mathrm{N}_v,\ \mathsf{ad})$, such that the received message differs or that the verification holds for an identity other than U_v (a case already addressed in Game 2). However, that in turn implies ability to create an adversary, $\mathcal{B}$ against the strong unforgeability of the signature. Thus,

$$\mathrm{Adv}_{\mathcal{A}}^{2-recv} \leq \mathrm{Adv}_{\mathcal{B}}^{SUF-CMA}(\lambda) \quad .$$

Case 2: Sender accepts maliciously. At some point U_v receives a flow of the form $\mathrm{N}_w,\ \mathsf{ad},\ \mathrm{Sign}_{sk_w}(\mathrm{U}_v,\ \mathrm{N}_v,\ \mathrm{U}_w,\ \mathrm{N}_w,\ \mathsf{ad})$, (with associated data corresponding to the components of either $\{\mathrm{Seq}_s\}_w$ or NACK). U_v checks the validity of the signature under the public signing key of U_w and accepts maliciously with $\mathsf{pid} = \mathrm{U}_w$ if U_w did not send a message of the form $\mathrm{N}_w,\ \mathsf{ad},\ \mathrm{Sign}_{sk_w}(\mathrm{U}_v,\ \mathrm{N}_v,\ \mathrm{U}_w,\ \mathrm{N}_w,\ \mathsf{ad})$ (all other variables under the signature are known to U_v). This in turn implies ability to create an adversary, $\mathcal{B}$ against the strong unforgeability of the signature, and thus

$$Adv_{\mathcal{A}}^{2-send} \leq Adv_{\mathcal{B}}^{SUF-CMA}(\lambda) \quad .$$

□

5.3 COCO-SYNC-P Protocol Analysis

As in Section 5.2, we restrict analysis of COCO-SYNC-P to the core protocol, i.e., the flows depicted in blue in the protocol description.

Theorem 5.3 (Security of COCO-SYNC-P)**.** *Let $\mathcal{A}$ be a PPT adversary against the* COCO-SYNC-P *security of the COCO-SYNC-P protocol* Π*, running time* $t' \approx t$*. Then*

$$Adv_{\Pi,\mathcal{A}}^{\textsf{COCO-SYNC-P}}(\lambda) = q^2 \cdot 2^{-\lambda} + 2n_{\mathcal{I}}^2 n_s^2 \cdot Adv_{\mu,\mathcal{B}(\lambda)}^{SUF-CMA}$$

where q is a polynomial bound on the number of queries allowed to $\mathcal{A}$, $n_{\mathcal{I}} = |\mathcal{I}|$ is the number of identities, $n_s = |[s]|$ the maximum number of sessions at an identity, and μ is a signature scheme.

The proof also follows similarly to that of COCO-SYNC-R, with the exception that we no longer require a *unique* partner for the requester, per the COCO-SYNC-P experiment.

5.4 RF Analysis

We will now analyze the RF footprint of the three protocols presented in this paper. For each protocol, we consider the worst-case analysis of the RF footprint under a correct protocol run, i.e., the most expensive run of the protocol where the protocol still terminates, and there has been no adversarial interference. If an adversary were to e.g., drop packets through jamming, it could force the protocol to loop or restart, hence we consider such instances outside of the RF baseline analysis. It is important to note that the *average* cases for these protocols will have significantly lower RF footprint than the worst case; our upper bounds simply represent the maximum footprint possible. We also provide a best case analysis.
We proceed under the assumption that inter-unit communications are less RF-expensive than SATCOM communications to the back-end. Using this assumption, the analysis is performed by calculating the RF cost per transmission as the product of power level and message length, represented as

$$\text{RF} = (PowerLevel) \cdot (|message|)$$

Inter-unit transmissions will be considered low power, while transmissions to the back-end will be considered high power.

Variables

In addition to the variables used for the protocols in Section 2, we add the following notation:

- k: Number of forward units participating in the protocol.
- P_L: RF footprint associated with low power RF transmission.
- P_H: RF footprint associated with high power RF transmission.
- ν: The field bit-length allotted for entity identifiers, $\mathrm{U}_v \in \mathcal{I}$.
- μ: The field bit-length allotted for a sequence number, i.e. $|\mathrm{Seq}_n|$. We assume that the field length allotted for the back-end sequence number (i.e., the sequence number Update), the nonce field length, and the NACK length are equivalent to the field length allotted for packet sequence numbers, i.e., $\mu = |\mathrm{Seq}_n| \approx |\mathrm{Update}| \approx |\mathrm{N}_v| \approx |\mathrm{NACK}|$.
- ρ: The length in bits of the signature tag generated from signing a message.
- ω: The length in bits of the update message M, i.e., $\mathrm{Seq}_{total} \cdot |M_n| = \omega$.

The analysis can be adjusted accordingly for cases where $|\mathrm{Seq}_n| \approx |\mathrm{Update}| \approx |\mathrm{N}_v|$ does not hold. For simplicity we assume that $\nu \leq \mu$.

BAC – Worst Case.
The worst case scenario for the BAC protocol occurs if the back-end transmits the update and none of the units receive any part of the update. In this case, one unit will reply with a confirmation message that indicates that all units are missing all packets.

Worst case We calculate the example worst-case RF footprint of the forward units as:

$$\begin{aligned}
\mathrm{RF}_{BAC} &\leq P_H \cdot \Big(|\mathrm{U}_v| + |\mathrm{N}_v| + |\mathrm{Update}| + k\big(|\mathrm{U}_v| + |\mathrm{N}_v| + \mathrm{Seq}_{total} \cdot |\mathrm{Seq}| + |sig|\big) \\
&\quad + |sig| \Big) \\
&\leq P_H \cdot \Big(\nu + 2\mu + k(\nu + \mu + \mu \cdot \mathrm{Seq}_{total} + \rho) + \rho \Big) \\
&\approx P_H \cdot \Big((k+1)(\nu + \rho) + \mu(k + k\mathrm{Seq}_{total} + 2) \Big) \ .
\end{aligned}$$

- In this worst-case scenario, the reply RF footprint expense of the confirmation assumes that each unit's confirmation message includes a request set of all sequence numbers, i.e., that none of the Update message itself was received. This gives us k multiples of $\text{Seq}_{total} \cdot \mu$. Naturally, most cases will have better performance, reducing or nearly nullifying the factor of Seq_{total}.

- The confirmation message also includes a unit identifier, nonce, update number, and signature tag, giving us k multiples of $(|\text{U}_v| + |\text{N}_v| + |\text{Update}| + \rho)$ where the field length allotted to a variable U_v is denoted $|\text{U}_v|$.

BAC – Best Case.
Now we calculate the best case RF footprint for BAC. This corresponds to when all packets are correctly received the back-end transmission.

$$\begin{aligned}
\text{RF}_{BAC} &\leq P_H \cdot \Big(|\text{U}_v| + |\text{N}_v| + |\text{Update}| + k\big(|\text{U}_v| + |\text{N}_v| + |\text{Seq}| + |sig|\big) \\
&\quad + |sig|\Big) \\
&\leq P_H \cdot \Big(\nu + 2\mu + k(\nu + \mu + \mu + \rho) + \rho\Big) \\
&\approx P_H \cdot (k+1)(\nu + 2\mu + \rho)
\end{aligned}$$

COCO-SYNC-R – Worst Case.
For the COCO-SYNC-R protocol, the worst-case RF footprint occurs if no unit received the full Update message, thus necessitating each requester to contact every other unit. Simultaneously, this occurs when the units cumulatively possess the full Update message, as otherwise less packets exist to share. In this scenario, each unit, when acting as U_v, must make a request to all other units and will receive the full Update message. In the worst case scenario, U_v only obtains the full Update message after sending requests to all other units – without loss of generality we simplify representation of this case by having $k-1$ units missing $\text{Seq}_{total} - 1$ packets, while the last unit is missing 1 packet.

Observing case Here we calculate based on that all units hold a given packet except for the last, which holds all packets but the specific one. We assume that units listen in (observing) on traffic not addressed to them:

$$\begin{aligned}
\mathrm{RF}_{SYNC-R} \leq P_L \cdot \Big(& (k-1) \cdot (|\mathrm{U}_w| + |\mathrm{N}_v| + |\mathrm{Update}| \\
& + (\mathrm{Seq}_{total} - 1) \cdot |\mathrm{Seq}| + |sig|) \\
& + (k-2) \cdot (|\mathrm{N}_w| + |\mathrm{NACK}| + |sig|) \\
& + (|\mathrm{N}_w| + (\mathrm{Seq}_{total} - 1) \cdot |\mathrm{Seq}| + |sig|) \\
& + (\mathrm{Seq}_{total} - 1) \cdot (|\mathrm{Update}| + |\mathrm{Seq}| + |\mathrm{Seq}| + |M_n| + |sig|) \\
& + (|\mathrm{U}_w| + |\mathrm{N}_v| + |\mathrm{Update}| + |\mathrm{Seq}| + |sig|) \\
& + (|\mathrm{N}_w| + |\mathrm{Seq}| + |sig|) \\
& + (|\mathrm{Update}| + |\mathrm{Seq}| + |\mathrm{Seq}| + |M_n| + |sig|) \\
& + k(|\mathrm{U}_v| + |\mathrm{N}_v| + |\mathrm{Update}| + |\mathrm{Seq}| + |sig|)\Big) \\
\leq P_L \cdot \Big(& (k-1) \cdot (\nu + 2\mu + \mu(\mathrm{Seq}_{total} - 1) + \rho) \\
& + (k-2) \cdot (2\mu + \rho) \\
& + (\mu \cdot \mathrm{Seq}_{total} + \rho) \\
& + (\mathrm{Seq}_{total} - 1) \cdot (3\mu + |M_n| + \rho) \\
& + (\nu + 3\mu + \rho) + (2\mu + \rho) + (3\mu + |M_n| + \rho) \\
& + k(4\mu + \rho)\Big) \\
\approx P_L \cdot \Big(& k\nu + 7k\mu + k\mu\mathrm{Seq}_{total} + 2k\rho + 4\rho + 3\mu\mathrm{Seq}_{total} \\
& + \rho\mathrm{Seq}_{total} + \omega\Big) \;.
\end{aligned}$$

- When the first unit takes its turn, it will request its missing sequence numbers from every other unit in Flow 1 of COCO-SYNC-R. Since the last unit holds all ($\mathrm{Seq}_{\mathrm{total}}$ - 1) needed packets, the request set will remain the same for $k-1$ requests. Thus we have $(k-1) \cdot (|\mathrm{U}_w| + |\mathrm{N}_v| + |\mathrm{Update}| + (\mathrm{Seq}_{total} - 1) \cdot |\mathrm{Seq}| + |sig|)$.
- In response to the first unit's requests, all units between the first and last unit in the hierarchy will send NACKs, giving $(k-2) \cdot (|\mathrm{N}_w| + |\mathrm{NACK}| + |sig|)$.
- Once the final unit receives the request from the first unit, it commits its ($\mathrm{Seq}_{\mathrm{total}}$ - 1) packets from cache. The result is $(|\mathrm{N}_w| + (\mathrm{Seq}_{total} - 1) \cdot |\mathrm{Seq}| + |sig|)$. Then, it transmits the packets it has committed to, resulting in $(\mathrm{Seq}_{total} - 1) \cdot (|\mathrm{Update}| + |\mathrm{Seq}| + |\mathrm{Seq}| + |M_n| + |sig|)$.

- The last unit (based on the original hierarchy) will request its missing packet, receive a confirmation of send, and then receive the packet transmission itself, resulting in $(|\mathrm{U}_w|+|\mathrm{N}_v|+|\mathrm{Update}|+|\mathrm{Seq}|+|sig|)+(|\mathrm{N}_w|+|\mathrm{Seq}|+|sig|)+(|\mathrm{Update}|+|\mathrm{Seq}|+|\mathrm{Seq}|+|M_n|+|sig|)$.

- Finally, all units, on their respective turns, will broadcast a confirmation message $k\cdot(|\mathrm{U}_v|+|\mathrm{N}_v|+|\mathrm{Update}|+|\mathrm{Seq}|+|sig|)$.

Non-observing case Here we calculate based on that all units hold a given packet except for the last, which holds all packets but the specific one. We assume that units do not listen in (non-observing) on traffic not addressed to them:

$$\begin{aligned}
\mathrm{RF}_{SYNC-R} \le P_L \cdot \Bigg(& \frac{(k-1)\cdot(k-2)}{2}\cdot(|\mathrm{U}_w|+|\mathrm{N}_v|+|\mathrm{Update}| \\
&+(\mathrm{Seq}_{total}-1)\cdot|\mathrm{Seq}|+|sig|) \\
&+\frac{(k-2)\cdot(k-3)}{2}\cdot(|\mathrm{N}_w|+|\mathrm{NACK}|+|sig|) \\
&+(k-1)\cdot(|\mathrm{N}_w|+(\mathrm{Seq}_{total}-1)\cdot|\mathrm{Seq}_n|+|sig|) \\
&+(k-1)\cdot(\mathrm{Seq}_{total}-1)\cdot(|\mathrm{Update}|+|\mathrm{Seq}|+|\mathrm{Seq}|+|M_n|+|sig|) \\
&+(|\mathrm{U}_w|+|\mathrm{N}_v|+|\mathrm{Update}|+|\mathrm{Seq}|+|sig|) \\
&+(|\mathrm{N}_w|+|\mathrm{Seq}|+|sig|) \\
&+(|\mathrm{Update}|+|\mathrm{Seq}|+|\mathrm{Seq}|+|M_n|+|sig|) \\
&+k\cdot(|\mathrm{U}_v|+|\mathrm{N}_v|+|\mathrm{Update}|+|\mathrm{Seq}|+|sig|)\Bigg) \\
\le P_L \cdot \Bigg(& \frac{(k-1)\cdot(k-2)}{2}\cdot(\nu+2\mu+\mu(\mathrm{Seq}_{total}-1)+\rho) \\
&+\frac{(k-2)\cdot(k-3)}{2}\cdot(2\mu+\rho) \\
&+(k-1)\cdot(\mu\cdot\mathrm{Seq}_{total}+\rho) \\
&+(k-1)\cdot(\mathrm{Seq}_{total}-1)\cdot(3\mu+|M_n|+\rho) \\
&+(\nu+3\mu+\rho)+(2\mu+\rho)+(3\mu+|M_n|+\rho) \\
&+k(4\mu+\rho)\Bigg) \\
\approx P_L \cdot \Bigg(& \frac{1}{2}k\Big(k\nu+3k\mu+k\mu\mathrm{Seq}_{total}+2k\rho-3\nu+5\mu-6\rho \\
&+8\mu\mathrm{Seq}_{total}+2\rho\mathrm{Seq}_{total}\Big)-\frac{7}{2}\mu\mathrm{Seq}_{total}+3\nu+25\mu+11\rho
\end{aligned}$$

$$+ k\omega - (k + \text{Seq}_{total} - 2)|M_n| - \rho\text{Seq}_{total}\Big)$$

- Each unit takes its turn requesting missing sequence numbers from every other unit in Flow 1 of COCO-SYNC-R. Each unit will request $\text{Seq}_{total} - 1$ from every other unit until reaching the unit with holding the packets. After its turn, each unit is placed at the bottom of the hierarchy, so the number of requests shrinks by one for each turn, resulting in $(k-1) + (k-2) + \ldots + 1$ requests for $\text{Seq}_{total} - 1$ packets. Thus we have $\frac{(k-1)\cdot(k-2)}{2} \cdot (|\text{U}_w| + |\text{N}_v| + |\text{Update}| + (\text{Seq}_{total} - 1) \cdot |\text{Seq}| + |sig|)$.

- All units except the unit with the $\text{Seq}_{total} - 1$ packets will NACK the request above, giving us the same series, except it begins at $k - 2$, since the final unit will not NACK. Thus we have $\frac{(k-2)\cdot(k-3)}{2} \cdot (|\text{N}_w| + |\text{NACK}| + |sig|)$.

- The final unit (based on the original hierarchy) will commit to sending $k - 1$ units $\text{Seq}_{total} - 1$ packets, i.e., $(k-1) \cdot (|\text{N}_w| + (\text{Seq}_{total} - 1) \cdot |\text{Seq}_n| + |sig|)$.

- If the unit commits to providing sequence numbers, it will provide the cache message from the BAC protocol to each unit that requested them, i.e., $(k-1)\cdot(\text{Seq}_{total}-1)\cdot(|\text{Update}|+|\text{Seq}|+|\text{Seq}|+|M_n|+|sig|)$.

- The last unit will request its missing packet, receive a confirmation of it, and then receive the packet transmission, i.e., $(|\text{U}_w| + |\text{N}_v| + |\text{Update}| + |\text{Seq}| + |sig|) + (|\text{N}_w| + |\text{Seq}| + |sig|) + (|\text{Update}| + |\text{Seq}| + |\text{Seq}| + |M_n| + |sig|)$

- Finally, all units, on their respective turns, will broadcast a confirmation message $k \cdot (|\text{U}_v| + |\text{N}_v| + |\text{Update}| + |\text{Seq}| + |sig|)$. For this calculation we assume that either all units are listening to the final broadcast or, more commonly, that the confirmation messages are sent to a designated unit for collection and transmission to the back-end. If neither of these cases are possible (i.e., that the confirmation messages must be individually sent to each other unit in the group) then a factor of $k - 1$ on this term is incurred.

COCO-SYNC-R – Best Case.
The best-case scenario occurs when all units have correctly received the full Update transmission and share confirmation of that inter-unit.

$$\begin{aligned}\mathrm{RF}_{SYNC-R} &\leq P_L \cdot \Big(k \cdot (|\mathrm{U}_v| + |\mathrm{N}_v| + |\mathrm{Update}| + |\mathrm{Seq}| + |sig|)\Big)\\ &\leq P_L \cdot k(\nu + 3\mu + \rho)\end{aligned}$$

COCO-SYNC-P – Worst Case.
The worst case for the COCO-SYNC-P protocol (run until matching transcripts among the receivers are achieved) occurs if no one unit received all Update packets from the initial BAC transmission. In this scenario, the maximum number of messages would need to be transmitted. Without loss of generality, we suppose that each unit has received non-empty set of packets $\{\mathrm{Seq}_t\}_v$ such that $|\{\mathrm{Seq}_t\}_v| = |\{\mathrm{Seq}_s\}_u| = q$ for all u, s and $\{\mathrm{Seq}_t\}_v \cap \{\mathrm{Seq}_s\}_u = \emptyset$. Thus $|\bigcup\limits_{u,s}\{\mathrm{Seq}_s\}_u| = \mathrm{Seq}_{total}$. In this calculation we assume that the protocol repeats until all units have broadcast their packets.

$$\begin{aligned}\mathrm{RF}_{SYNC-P} \leq P_L \cdot \sum_{i=0}^{k} \Bigg(& |\mathrm{N}_w| + |\mathrm{Update}| + |sig|\\ & + \Big(|\mathrm{U}_u| + |\mathrm{N}_u| + |\mathrm{Update}| + |\mathrm{Seq}| \cdot |\{\mathrm{Seq}\}^*_u| \cdot \big(i(k-1)\\ & + (k-i)(k-1-i)\big) + |sig|\Big)\\ & + |\mathrm{N}_v| + |\mathrm{N}_w| + |\mathrm{Seq}| \cdot |\{\mathrm{Seq}\}_v| + |sig|\\ & + |\{\mathrm{Seq}\}_v| \cdot (|\mathrm{Update}| + |\mathrm{Seq}| + |\mathrm{Seq}_{total}| + |M_n| + |sig|)\Bigg)\\ & \leq P_L \cdot \Big(2k\mu + k\rho\\ & + k\nu + 2k\mu + \mu \cdot q \cdot \sum_{i=0}^{k}(k^2 - k - ik + i) + \rho + 2k\mu + k\mu \cdot q + k\rho\\ & + kq \cdot (3\mu + |M_n| + \rho)\\ & \approx P_L \Big(k\nu + 6k\mu + 2k\rho + \rho + 4kq\mu + kq|M_n| + kq\rho\\ & + q\mu(\frac{1}{2}k^3 + k^2 - \frac{3}{2}k)\Big)\\ & \approx P_L \Big(k\nu + 6k\mu + 2k\rho + \rho + \frac{5}{2}\mu\mathrm{Seq}_{\mathrm{total}} + \omega + \rho\mathrm{Seq}_{\mathrm{total}}\\ & + \frac{1}{2}k^2\mu\mathrm{Seq}_{\mathrm{total}} + k\mu\mathrm{Seq}_{\mathrm{total}}\Big)\end{aligned}$$

- To initiate the protocol, a unit from {U} must send Flow 1, consisting of a nonce, the update number, and a signature. This gives us $|\mathrm{N}_w| + |\mathrm{Update}| + |sig|$.
- Once the protocol has been initiated, each unit then broadcasts its request set. The broadcast request consists of a nonce, update number, the set of requested sequence numbers, and a signature. The request set decreases with each protocol iteration, giving us $\Big(|\mathrm{U}_u| + |\mathrm{N}_u| + |\mathrm{Update}| + |\mathrm{Seq}| \cdot |\{\mathrm{Seq}\}_u^*| \cdot \big(i(k-1) + (k-i)(k-1-i)\big) + |sig|\Big)$.
- Following the requests, one unit will be set as Lead. Since all units will have the same size request set, the role of Lead will fall to the unit highest in the hierarchy. Throughout the run of the protocol, Seq_{total} sequence numbers will eventually be provided. This gives us $|\mathrm{N}_v| + |\mathrm{N}_w| + |\mathrm{Seq}| \cdot |\{\mathrm{Seq}\}_v| + |sig|$.
- The lead unit then provides its commit set by relaying the cached messages from the back-end. This gives us $|\{\mathrm{Seq}\}_v| \cdot (|\mathrm{Update}| + |\mathrm{Seq}| + |\mathrm{Seq}_{total}| + |M_n| + |sig|)$.
- The above sequence repeats for all units, with the lead in each round possessing an increased packet set by a term of q.

COCO-SYNC-P – Best Case.
As with COCO-SYNC-R, the best-case scenario for COCO-SYNC-P occurs when all units have correctly received the full Update transmission and share confirmation of that inter-unit.

$$\begin{aligned}\mathrm{RF}_{SYNC-P} &\leq P_L \cdot \Bigg(|\mathrm{N}_w| + |\mathrm{Update}| + |sig| \\ &\quad + k\Big(|\mathrm{U}_v| + |\mathrm{N}_u| + |\mathrm{Update}| + |\mathrm{Seq}| + |sig|\Big)\Bigg) \\ &\leq P_L \cdot \Big(2\mu + \rho + k(\nu + 3\mu + \rho)\Big)\end{aligned}$$

6 Conclusion

We have provided three protocols for use in a constrained RF scenario: the BAC protocol which provides a means for a remote back-end to provide an update to forward units while minimizing the amount of data that is sent

by the forward units over SATCOM, the COCO-SYNC-R protocol which allows units to share update data among themselves via a pull method, and COCO-SYNC-P which utilizes a push approach for inter-unit sharing. These protocols detail a robust method for updating EABO software (or sharing a mutual view of other information) and provide options to the end user on how to share packets amongst receivers.
We provide security models and analysis to show assumptions under which these protocols are secure. Additionally, we mathematically represent bounds for worst case RF scenarios for all three protocols, based on the forward units' RF footprint.
Future work opportunities include implementation of the protocols described above and optimizing the performance of the protocols in sparsely connected networks. Additionally, extensive testing will need to be performed to determine which variation of the COCO-SYNC sharing protocols is best suited for specific scenarios that result from varying environments and varying physical layer transmission for inter-unit communication links.

References

[1] General David H. Berger. Commandant's Planning Guidance. *USMC*, 1(1):1–26, 10 2020.

[2] Imanol Mugarza, Jose Luis Flores, and Jose Luis Montero. Security Issues and Software Updates Management in the Industrial Internet of Things (IIoT) Era. *Sensors (Basel, Switzerland)*, 20(24):7160–, 2020.

[3] Luigi Catuogno, Clemente Galdi, and Giuseppe Persiano. Secure Dependency Enforcement in Package Management Systems. *IEEE Transactions on Dependable and Secure Computing*, 17(2):377–390, 2020.

[4] Lukas Zapolskas and Nicolas Gailly. Increasing Software Update Security Through PGP-Compatible Threshold Signatures, Nov 2021.

[5] Solon Falas, Charalambos Konstantinou, and Maria K Michael. A Modular End-to-End Framework for Secure Firmware Updates on Embedded Systems. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(1):1–19, 2021.

[6] Juan Benet. IPFS – Content Addressed, Versioned, P2P File System, 2014.

[7] United States Marine Corps. *COMMUNICATION EQUIPMENT B191716 STUDENT HANDOUT*. United States Marine Corps.

[8] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT Key Exchange with Full Forward Secrecy. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 519–548. Springer, 2017.

[9] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of tls-dhe in the standard model. In *Advances in Cryptology – CRYPTO 2012*, pages 273–293. Springer Berlin Heidelberg, 2012.

[10] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO' 93*, pages 232–249. Springer Berlin Heidelberg, 1994.

[11] Britta Hale and Colin Boyd. Computationally Analyzing the ISO 9798-2.4 Authentication Protocol. In *SSR*, 2014.

[12] Cas Cremers, Britta Hale, and Konrad Kohbrok. The Complexities of Healing in Secure Group Messaging: Why {Cross-Group} Effects Matter. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1847–1864, 2021.

[13] Fagen Li, Hui Zhang, and Tsuyoshi Takagi. Efficient Signcryption for Heterogeneous Systems. *IEEE Systems Journal*, 7(3):420–429, 2013.

7 Appendix

7.1 Primitive Definitions

Definition 7.1 (Digital Signature Scheme)**.** *(pulled from [12])* μ *is an allowed digital signature scheme if* $\mu = (\mu.Gen, \mu.Sign, \mu.Verify)$ *such that:*

- $\mu.Gen$ *is a probabilistic key generation algorithm which takes as input a security parameter* $\lambda \in \mu$ *and returns a pair (sk, pk), where sk is the secret key and pk is the public key.*

- $\mu.Sign$ *is a (possibly probabilistic) signing algorithm which takes as input the secret key sk and a message* m *and returns* σ*, a signature on* m*.*

- *μ.Verify is a deterministic verification algorithm which takes as input a public key pk, a message m, and a signature σ_m and outputs bit* verify *such that* verify *= 1 if σ_m is a valid signature on m and* verify *= 0 otherwise.*

Definition 7.2 (EUF-CMA)**.** *(adapted from [13]) A digital signature scheme is said to have the EUF-CMA property if no polynomially bounded adversary has a non-negligible advantage in the following game.*

Initialize: *The challenger $\mathcal{T}$ runs μ.Gen algorithm to generate (sk, pk) and provides the public key pk to adversary $\mathcal{F}$. $\mathcal{F}$ is provided with a signing oracle $\pi_{\mathcal{E}}$ which accepts message m and returns σ_m such that σ_m = μ.Sign(sk, m).*

Attack: *$\mathcal{F}$ performs a polynomially bounded number of queries to $\pi_{\mathcal{E}}$ with $m \in \{m_0,\ m_1,...,m_q\}$ and receives back the corresponding σ_m from $\pi_{\mathcal{E}}$.*

Forgery: *$\mathcal{F}$ achieves forgery if he can generate (m^*, σ^*) such that $m^* \notin \{(m_0,\ m_1,...,m_q\}$ and μ.Verify(pk, $(m^*, \sigma^*)) = 1$.*

Definition 7.3 (SUF-CMA)**.** *(adapted from [13]) A digital signature scheme is said to have the SUF-CMA property if no polynomially bounded adversary has a non-negligible advantage in the following game.*

Initialize: *The challenger $\mathcal{T}$ runs μ.Gen algorithm to generate (sk, pk) and provides the public key pk to adversary $\mathcal{F}$. $\mathcal{F}$ is provided with a signing oracle $\pi_{\mathcal{E}}$ which accepts message m and returns σ_m such that σ_m = μ.Sign(sk, m).*

Attack: *$\mathcal{F}$ performs a polynomially bounded number of queries to $\pi_{\mathcal{E}}$ with $m \in \{m_0,\ m_1,...,m_q\}$ and receives back the corresponding σ_m from $\pi_{\mathcal{E}}$.*

Forgery: *$\mathcal{F}$ achieves forgery if he can generate (m^*, σ^*) such that $(m^*, \sigma^*) \notin \{(m_0, \sigma_0),\ (m_1, \sigma_1),...,(m_q, \sigma_q)\}$ and μ.Verify(pk, $(m^*, \sigma^*)) = 1$.*

7.2 Decision Points

Here, we discuss some decision points and trade-offs that arose throughout the development of these protocols. Some points became more pertinent than others and may have been discussed in the main body of the paper. The purpose of this section is to provide background on the variety of decision points.

A first decision point is whether authentication should be done with symmetric or asymmetric keys. Specifically, some options include key encapsulation to the forward units followed by symmetric authentication, vs. digital signatures. Symmetric authentication would result in shorter message transmissions (i.e., from a shorter tag length), but come at the cost of (a) a requirement that all parties received their respective encapsulated keys and can thus verify the transmission (assuming individually key encapsulations), (b) lack of back-end authenticity for packets distributed among forward units (if a group key is encapsulated). In the case of (b), an adversary could compromise all inter-unit communications and packet sharing if it just compromised one unit and obtained the transmission key. Ultimately, digital signatures proved to be a better fit given the requirements of the situation, particularly with respect to resiliency to packet dropping.

Another decision point occurs when receiving units request missing packets. The first option would be to request the packet from the original sender. This would significantly spotlight that unit's location but would not involve the other receiving units. The other option would be to rely on inter-unit communications to share missing packets. This method is anticipated to generate significantly less RF because SATCOM is not used, but would spotlight all units to a lesser extent. We chose to use inter-unit communications in COCO-SYNC-R and COCO-SYNC-P.

The inter unit communications can be done in multiple ways. The two methods presented here represent two options of push and pull mechanisms. The pull method (COCO-SYNC-R) relies on a predetermined communications hierarchy amongst the units, requests for missing packets, NACKs, and timeouts as a failsafe. This method is more complex in practice than the alternative, but it does not spotlight a single unit's location as transmissions are spread evenly, and the hierarchy can be set to minimize location sharing for certain units. The push method (COCO-SYNC-P) relies on all units sharing packet requests and the unit with the least number of needed packets taking a lead role in sending to other units. This method is less complex in practice and potentially more efficient in sharing time, but introduces unnecessary packet transmission and may to a greater extent highlight units' locations.

Further, the push method itself can be conducted in two distinct ways. The method presented consists of every unit broadcasting their missing packets followed by a designated leader broadcasting all of its messages. The process repeats until all units have the same transcript. This manner offers the most basic form of communication for units to build on and is automatic in that the lead unit immediately sends the messages. An alternative option, not focused on here, would be for the other units to send the lead unit the messages it has requested, whereupon the lead would compile the most complete update and then broadcast. This technique is presents a middle ground between the presented COCO-SYNC-R and COCO-SYNC-P.

Yet another trade off considered is the balance between a requirement to limit RF signature at the receiving units or limit bandwidth requirements on the back-end. A simple approach which optimizes on the former is to send the Update on repeat for an extended period. This would maximize the probability that all messages are received by the forward units and would significantly limit the requirement for the forward units to communicate amongst each other. More importantly, it would reduce necessary transmissions to the back-end (which incur RF footprint on the forward units) as it gives multiple opportunities for all units to receive the original message. Furthermore, if RF footprint from unit transmission to the back-end is high-risk, the units may choose to delay or forgo the confirmation message for an extended period of time. The drawback to this option is the significant bandwidth incurred on the back-end. Given, however, that the back-end is likely e.g., a Marine Expeditionary Force (MEF) level asset, it is reasonable to assume the capability exists and could be used in such a manner. The alternative, of course, is to only send the Update once and leave the forward units to communicate amongst themselves, responding with potentially more requests to the back-end. Given the importance of limiting RF footprint, we focus on the prior option.