A new approach for information dissemination in distributed JISR coalitions

Christian Kerth[a], Philipp Klotz[b], and Barbara Essendorfer[c]

[a,b,c]Fraunhofer IOSB, Fraunhoferstrasse 1, Karlsruhe, Germany

ABSTRACT

To meet todays challenges in ISR (Intelligence, Surveillance and Reconnaissance) defense coalitions Systems of Systems (SOS) architectures are needed that are flexible, function in a networked environment and support relevant operational doctrine and processes. To enable the distributed production of intelligence in networked operations the Intelligence Cycle and Joint ISR (JISR) provide process descriptions that adhere to multinational and multisystem collaboration. An interoperable SOS architecture supporting those processes needs to make use of standards for data/information management with a special focus on dissemination.

The NATO ISR Interoperability Architecture (NIIA) and supportive standards (STANAGS- standardization agreements) have been specified to provide a solution to these needs. In terms of data distribution, STANAG 4559 is the core standard of relevance here. It defines a concept, data- and information models, interfaces and services to support information dissemination according to JISR. The current specification for synchronization of JISR results however has some deficiencies in terms of implementation complexity, flexibility, robustness and performance. Thus, there is a need for a new approach to data dissemination in networks implementing STANAG 4559 that enables the usage of all aspects currently supported by this standard but seeks to solve the known issues.

Thereupon this paper presents requirements for data dissemination in a JISR enterprise, derives key performance indicators (KPIs), identifies possible technical approaches and finally defines a new solution based on the concept of Hash Tries. Here a tree-based data structure is organized based on hashes of nodes, which allows a quick identification of changes in replicated data.

Keywords: ISR, CSD, STANAG 4559, Joint ISR, Hash Tries

1. INTRODUCTION

In military operations intelligence is derived through a complex process from information collected by surveillance and reconnaissance assets. Within NATO, UNO or EU missions multiple nations cooperate, and bring their assets and systems to the country of deployment. They also might connect them with locations back in their respective home country, where information might be processed. The systems and services provided by the different nations were normally developed independently from each other, possibly by different vendors and based on diverging requirements. Also different security domains might be involved, and network availability is not always given especially if mobile units (of navy or army) are connected. Still it is important that the information is exchanged in time, and that relevant information is disseminated to the right person to be able to make a well-informed decision.

To enable information exchange in such a complex environment, where systems, services and consequently acting persons need to be interoperable, it is necessary to have processes defined, that describe what is exchanged and who is involved with which responsibility. The Intelligence Cycle (Intel Cycle)[1] and the Joint ISR (Intelligence, Surveillence and Reconnaissance) process[2] were defined in this respect. These processes need to be picked up by a system architecture that enables the integration of systems and services. In terms of data and

Further author information:
Christian Kerth: E-mail: christian.kerth@iosb.fraunhofer.de
Philipp Klotz: E-mail: philipp.klotz@iosb.fraunhofer.de
Barbara Essendorfer E-Mail: barbara.essendorfer@iosb.fraunhofer.de

information processing this implies that data is marked, so that it can be processed and passed on adequately. As systems from different nations and vendors are involved, there need to be coordinated data formats and information models so that the syntax as well as the content can be understood independently from the system or service used. To connect those systems and services (interoperable) interfaces have to be defined.

The NATO ISR Interoperability Architecture (NIIA)[3] and supportive standards (STANAGS- standardization agreements) have been specified to provide a solution to these needs. Within the NIIA also the connection between those standards is defined, as well as the mapping of data elements used within the different STANAGs. The NIIA can be seen as a high-level framework architecture that needs to be tailored depending on the specific setting. A core element of the NIIA is STANAG 4559[4] that focusses on data and information dissemination (mainly) within the Joint ISR (JISR) process. As every standard this STANAG is a living document that has been developed in a collaborative effort by different nations within a Custodian Support Team (CST). It receives updates based on requirements from operational experience within deployment, trials and exercises as well as due to technological evolution.

Within this paper the JISR process, use cases and requirements are examined and aspects of STANAG 4559 are described in more detail. Then a deficit presumed in STANAG 4559 is explored and a possible solution based on the concept of Hash Tries[5] is described. The paper closes with an overview and possible next steps.

2. JISR COALITIONS

2.1 Operational context and processes

In defense coalitions, where different nations cooperate and multiple systems have to be incooperated into a system of systems (SOS) architecture, it is of utmost importance to understand which processes need to be supported. This defines how systems need to be connected by dedicated interfaces and formats. Intelligence is achieved by the ability to combine information from different intelligence disciplines (as Imagery Intelligence-IMINT, Signal Intelligence-SIGINT or Open Source Intelligence-OSINT). The information itself is produced within the JISR process,[2] where based on a specific validated *collection requirement (CR) tasks* are defined and assigned to a specific collection asset. Based on that the asset produces a *collection product* that is then passed on to a process and exploit(ation) service (or system). Here the *JISR result* is produced that is fed back into the Intel Cycle (also see[1]). Within the Intel Cycle *JISR results* from multiple sources and potentially different intelligence disciplines are then further processed and analyzed to lead to an overall situation awareness. These processes and procedures are described in more detail otherwhere.[6] The operational settings, hierarchies and assignments of systems and services are normally defined within an *Order of Battle (ORBAT)* and are specific to a mission.

2.2 Use cases for information dissemination

Based on the processes defined above different use cases for information dissemination within a JISR coalition can be of interest. Depending on the operational setting and the specific mission the network connection and performance, the security domains, the system assignments as well as the overall information flow differs. Thus the requirements for a SOS architecture differ depending on the use case. The JISR process is normally supported by a *"federated and collective effort from all levels of command, across components, and possibly supported by national and/or out-of-the-theatre capabilities"*.[2]

Information dissemination within a unit and possibly one network node that is located in one security domain is the use case with the least complexity. Information dissemination here can be performed within a local area network (LAN) without security gateways or release servers, and the systems supporting the processes are assigned to one unit although possibly provided by different nations and vendors. This implies that the information elements (as tasks, requests or JISR results) should be standardized as well as the interfaces and/or services to exchange information between the systems.

Once different units are connected over a wide area network (WAN) with possibly poor performance and maybe located in different security domains, it is required to adhere to strict techniques that spare bandwidth and that enable need based information dissemination. Additionally security devices like gateways or release servers have to be integrated and security standards have to be respected. A specific issue here is the need

to be able to delete data due to security restrictions or privacy aspects (such as defined within EUs General Data Protection Regulation (GDPR)[7]). Doing this within an enterprise of distributed services implies a specific challenge.

In a complex enterprise where *processing, exploitation and dissemination (PED)* is executed at a different location than *collecting* additional aspects as low bandwidth connections, arbitrary network topologies and assets/capabilities that connect and disconnect come into focus and need to supported by a SOS architecture.

Within the next chapter a current solution to support JISR within use cases like described above is described, and a closer look is taken at information dissemination within a complex enterprise.

3. DATA DISSEMINATION WITH THE CSD CONCEPT

3.1 STANAG 4559

The STANAG 4559 aims to increase the interoperability between different systems and NATO partners for the dissemination of products like imagery, videos, reports, tasks, other IRM&CM (Intelligence Requirement Management and Collection Management) artifacts and other documents. Therefore the architecture, use cases, process and information models, services and standard interfaces for querying and accessing different product types are defined and described inside this standard. The whole concept is called the Coalition Shared Data (CSD) concept.

As described in the previous chapter, different products can be created during the JISR process. After the task-based creation, the product is stored according to user requirements and network guidelines and can be shared with other people or communities. A detailed process description can be found in.[6]

The standard talks about three different product types:

- static products
- dynamic products
- stream products

Since the ratification of Edition 4 the standard is divided into three specification documents called AEDP (Allied Engineering Documentation Publication) - one for each product type and the corresponding processes.

- AEDP-17: NATO STANDARD ISR LIBRARY INTERFACE[8]
  Dissemination of Static products via the CSD Server
- AEDP-18: NATO STANDARD ISR STREAMING SERVICES[9]
  Dissemination of Streaming products via the CSD Streaming Server
- AEDP-19: NATO STANDARD ISR WORKFLOW ARCHITECTURE[10]
  Dissemination of Dynamic products via the ISR Workflow Architecture

3.2 Dissemination of static products described in AEDP-17

To make static products and information available to all nodes in a network, the standard defines a dissemination concept for products and matching catalogue entries. Here principles and techniques how created products are handled and shared are defined.

After the creation of a product (for example by a sensor), the product itself and its additional metadata are stored inside the CSD-Server according to STANAG 4559 AEDP-17.[8] The catalogue entry represents the metadata providing information (location, creation date/time, security classification, etc.) about the product itself. The available catalogue entries (related to products) can be searched by clients and partly be changed or supplemented (on the originating CSD-Server). Therefore, the client system defines a query which returns results according to the used filter and query parameters. It is also possible to set up a continuous query. Thereby not

only the current stock of catalogue entries are considered but also catalogue entries that are added during the execution of the continuous query are taken in account. The user can then examine the query results, with some product types he can then also inspect related data like an overview (of an image). If the user is interested in the product itself he can then request and download it through the client.

To synchronize the products and catalogue entries between different CSD-Servers a synchronization procedure has to be set up. Therefore the standard mandates to use queries to search the content of the synchronization partner and request the catalogue entry if it is missing in the local system. To reduce network bandwidth, only the catalogue entry is synchronized automatically. The product itself (image, video, etc.) is normally only shared if requested by a (client) system. Thus, each system has a copy of each catalogue entry in its local storage.

Over the years the concept was already implemented and operationalized in different projects and issues and problems were identified and noticed in different communities. The following issues were identified when practicing the approach in a large enterprise:

- Problems with incorrect configuration

  There are a lot of parameters that need to be defined before the synchronization can be used, leading to a high possibility of incorrect configuration that affects (due to the synchronization concept) the whole enterprise.

  - Network configuration

    Configuration of the network components: For example firewalls, network topology and existing systems etc.
  - Synchronization configuration

    Configuration of the synchronization components: For example endpoints, used ports and authentication/authorization parameters etc.

- Flaws in synchronization concept

  Problems with the concept itself defined through the standard:

  - The usage of queries for synchronization is error prone.
  - While synchronizing it is necessary to have further information about the network topology of the coalition:
    * As it needs to be guaranteed that no product is missed, a CSD-Server system needs to know all its direct neighbors.
    * Also the timestamp of the last received product needs to be stored.
      This leads to problems when the network topology is changing or systems are currently not available (due to maintenance or network issues).
  - The interface to connect CSD-Servers with each other is defined in Common Object Request Broker Architecture (CORBA)[11]
    * CORBA is perceived as a technology that is phasing out (e.g. JAVA is no more supporting CORBA in its current LTS version)[12] .
    * CORBA is perceived to introduce issues if dealing with security gateways/ release servers.
  - Product dissemination between systems with different metadata models (STANAG versions) is, at least, difficult.
  - Deletion of products and catalogue entries is not supported and not easy to realize.

This is not an exhausitve list, but it should show some of the points that led us to the conception of a new and better way to determine standardized metadata and product files dissemination.

4. DEFINING A NEW APPROACH TO DATA DISSEMINATION

Based on the apects mentioned in the previous chapter, the new data dissemination approach focuses on simplifying the configuration to easen the work of operators, as well as resolving flaws in the synchronization concept itself. In order to do so, the new approach has to:

- Minimalize the amount of required configuration.
- Share Catalogue Entries in the entire enterprise, regardless of:
  - their metadata model,
  - where they were ingested / updated.
- Make files accessible from the entire enterprise, regardless of:
  - the (initially) storing service.
- In an environment of:
  - (ir)regular network disconnects,
  - low bandwidth,
  - arbitrary topology,
  - constant change in connected servers.

Synchronization in general addresses the replication of catalogue entries, file transfer, data model (extension) replication and replication of associations and related files. As a first step the further sections of this paper focus on the catalogue entry dissemination, to keep the scope within a reasonable amount.

4.1 Use cases and technical requirements

4.1.1 Use cases

Searches performed on the catalogue entries stored locally in a CSD-Server are considered outside the scope of dissemination. All catalogue entries are to be shared by all CSD-Servers and searches are performed using the catalogue entries stored in a CSD-Server (i.e. no federated search).

Thus, the following three use cases are of relevance:

As CSD-Servers join operational networks over time, one use case is to join an existing network, where catalogue entries are already stored. Also, the joining CSD-Server will potentially bring some catalogue entries. In a more generalized form, the use case can be expressed as 'when joining, multiple (previously disconnected) networks are connected and subsequently all catalogue entries from all networks need to be disseminated in the new, now larger network'.

Just as servers join the network during operations, CSD-Servers can also permanently leave the network, while the other CSD-Servers remain operational. The catalogue entries previously made available from the leaving server will remain available in the network.

In case of system failure, some of the lost data can be restored from backups. It is expected, that the new approach is able to recover catalogue entries that were previously disseminated to other CSD-Servers back from the network.

4.1.2 Requirements

A set of requirements was identified based on the uses cases and experience gathered from previous usages of connected CSD-Servers. Thus, in order to be useful in operations, the new approach needs to:

- support concurrent storage of catalogue entries in the range order of 100K (lower interval range of estimation) to 10M (upper interval range of estimation),
- support versioning of catalogue entries,[13]
- support dissemination if knowing only the surrounding network, i.e. no knowledge of the entire network topology,
- be robust against network disconnects and (remote) changes in network topology,
- work with available network bandwidths in the range 1Mbit to 10Mbit, possibly with network round trip times of 100msec to 500msec,
- support multiple data models at the same time,
- disseminate catalogues independent of the original ingest location.

4.2 Approaches/data structures

Data dissemination can be achieved using several approaches / data structures. The following list of candidates was further evaluated. We added the most important findings:

- Queries
  - current query mechanism lacks expressive power for change detection (cost to evolve standard)
  - high computational cost (lookup mechanisms are generally faster than queries/filters)
  - lack of strong resume guarantees (no resume tokens in queries)
  - diverse strategies for query structure (possible cause of interoperability issues)
- Lists (include patterns such as replay logs)
  - strong guarantee for resumes (using resume tokens)
  - low computational cost (lookup of next element is O(1)[14])
  - lack of hard delete if append only (problematic with GDPR[7])
  - weak support for partial data loss on consumer side (need to re-read all data in worst case)
- (binary) Hash Tries (as special case of tree structure)
  - strong guarantee for resumes (when used with checksums in nodes)
  - for large number of elements, every node organizes (with small deviation) the same number of elements as its sibling node. → O(log(n))[14] for read access
  - O(1)[14] cost to detect no change, O(log(n))[14] cost to identify change → scales well for large amounts of data
  - adding entries is more cost intensive than for lists, yet in practice low enough

With our requirements and use cases in mind, we evaluated that Hash Tries are the most fitting approach to take.

4.3 Hash Tries

The Hash Trie data structure was tweaked slightly to further improve its capabilities for us:

- All elements are added based on the hash of their content (SHA-384[15] in our case). This leads to a balanced data structure and performance improvement.
- If there is only one element in a branch left to organize, it is stored in the first node that organizes one element, instead of a long tail. This reduces the number of accesses to reach a catalogue entry in the Hash Trie, resulting in a performance improvement.
- All nodes in the Hash Trie are checksummed (also SHA-384[15]). This enables check identification if content of Hash Trie changed.

Elements organized in the Hash Trie can move up an down in the Hash Trie (depending other elements sharing the same hash prefix). However they can never move horizontally (as their checksum never changes). The resulting Hash Trie only depends on the data it organizes; it does not differ if elements are added in a different order.

Instead of using a binary tree, we could have used an octal tree or hexadecimal tree as well. However for fast access over the network this would require each node access to return the next set of characters that are organized below the current node. We chose not to do so to keep the size per node small.

Other hashing algorithms could have been used, however in practice SHA-384 turned out to be a good match for resilience against hash collisions and required speed.

In the example of figure 1, a Hash Trie contains three catalogue entries. Based on their checksums, they are placed in different branches of the tree. The catalogue entry colored yellow is the only one in the example that has the highest bit value '0' in the hash, thus it is the only catalogue entry placed in the left branch. Both the catalogue entry colored green and orange have the highest bit value of '1' in their hash, thus they are both placed in the right branch. With their next lower bit differing, the green catalogue entry is placed in the left sub-branch, whereas the orange catalogue entry is placed in the right sub-branch. All the nodes in the Hash Trie (depicted by blue circles) receive checksums based on the catalogue entries attached to them and the checksums of their child node, if they have any.

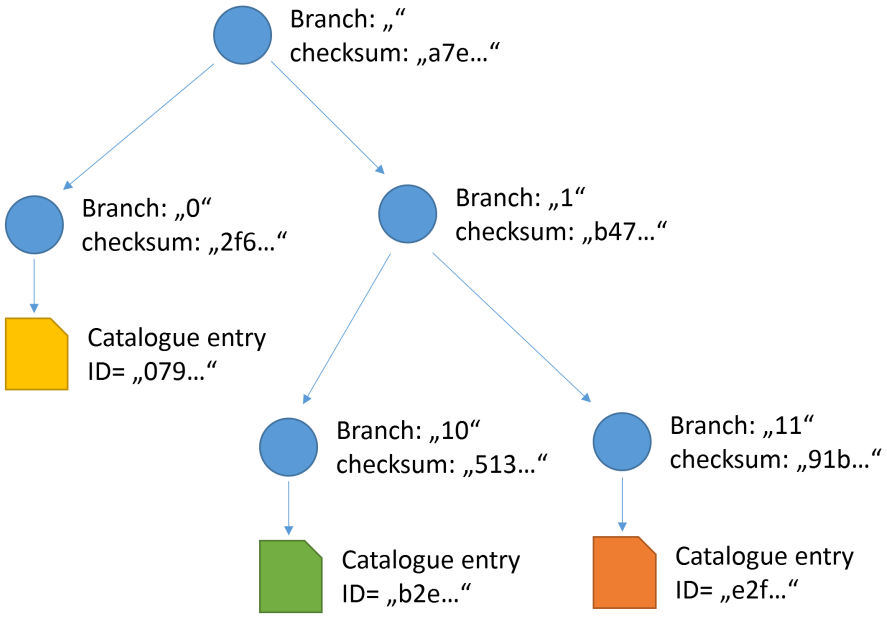

Figure 1. Example Hash Trie

4.4 Hash Tries for data dissemination

Within the network, all CSD-Servers organize their available catalogue entries in Hash Tries. Repeatedly, every CSD-Server compares his Hash Trie against the Hash Tries of adjacent CSD-Servers. If there are differences, the CSD-Server retrieves the element present in the remote server that do not exist in the local server and adds it to its own Hash Trie. Thus, an eventual consistency over time is achieved.

To compare Hash Tries:

- Start by comparing the root nodes.
- Compare the checksums of the current node:
  - If their checksums are the same, all data organized below is the same.
  - If their checksums differ:
    - If there are catalogue entries in the current node, compare the catalogue entries in the current node.
    - If there are no catalogue entries in the current node, compare the child nodes of the current node.

The algorithm above does result in consistency over time for adjacent servers. However, as catalogue entries can move up and down in the Hash Trie, it can happen that not all changes are identified in a single application of the algorithm. The algorithm was considered good enough for the use cases at hand, however using it for other use cases could lead to undesired delays in dissemination.

5. CONCLUSION

In this paper we introduced a new concept to synchronize data between CSD-Servers in JISR coalitions based on Hash Tries. The concept we chose gives strong resume guarantees and is able to handle data loss scenarios well. Also it should make configuration in big enterprises less complex and error prone and it makes use of current technologies.

We have prototypically implemented the concept and compared it with the current solution. The expected benefits were accomplished. We identified some aspects that could be improved in the next iteration.

- The current polling mechanism can be complemented with eventing between nodes (e.g. element added event) to reduce latency for dissemination.
- Currently the checksums of the Hash Trie remain different until elements added in remote system are added to the local Hash Tries. This means that data elements need to be retrieved before the next iteration to avoid elements being identified as added over and over again. A solution to improve that would be to have a concurrent check for changes and remote retrievals and/or concurrent checks on multiple servers without downloading the same element from different servers.
- At the moment tombstones [16, p. 159] can be used to delete data. Here it would also be of interest to examine if an appropriate approach exist that does not require them, as lots of catalogue entry deletions result in a large number of tombstones to accumulate.
- An automatic cleanup when elements are updated (i.e. a newer version is stored) might be useful in the long run.

The introduced concept is currently not part of the standard. Here a next step will be to introduce it in more detail to the CST and eventually reach a consensus.

ACKNOWLEDGEMENT

Parts of the work of the authors being described in this publication has been funded by the BMVg (Federal Ministry of Defence). The standard STANAG 4559 is developed within the CST of STANAG 4559. The authors acknowledge valuable help and contributions from all partners within the CST.

REFERENCES

[1] NATO Standardization Office (NSO), "AJP 2.1 ALLIED JOINT DOCTRINE FOR INTELLIGENCE PROCEDURES." https://nso.nato.int/nso/nsdd/APdetails.html?APNo=1976&LA=EN (2016). (Accessed: 15 March 2019).

[2] NATO Standardization Office (NSO), "AJP 2.7 ALLIED JOINT DOCTRINE FOR JOINT INTELLIGENCE, SURVEILLANCE AND RECONNAISSANCE." https://nso.nato.int/nso/nsdd/APdetails.html?APNo=1952&LA=EN (2016). (Accessed: 15 March 2019).

[3] NATO Standardization Office (NSO), "STANREC 4777 NATO INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE INTEROPERABILITY ARCHITECTURE." https://nso.nato.int/nso/nsdd/stanrecdetails.html?idCover=8591&LA=EN (2018). (Accessed: 15 March 2019).

[4] NATO Standardization Office (NSO), "STANAG 4559 - NATO STANDARD ISR LIBRARY INTERFACES AND SERVICES." https://nso.nato.int/nso/zPublic/stanags/CURRENT/4559EFed04.pdf (2018). (Accessed: 15 March 2019).

[5] Merkle, Ralph C., "A Digital Signature Based on a Conventional Encryption Function," in [*Advances in Cryptology — CRYPTO '87'*], Pomerance, Carl, ed., 369–378, Springer Berlin Heidelberg, Berlin, Heidelberg (1988).

[6] Essendorfer, B., Hoffmann, A. Sander, J. and Kuwertz, A., "Integrating coalition shared data in a system architecture for high level information management," in [*Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies II*], **10802**, 108020F, International Society for Optics and Photonics (2018).

[7] EUROPEAN PARLIAMENT AND COUNCIL, "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)." Official Journal of the European Union. http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN (2017). (Accessed: 18 March 2019).

[8] NATO Standardization Office (NSO), "NATO STANDARD ISR LIBRARY INTERFACE-AEDP-17." https://nso.nato.int/nso/nsdd/apdetails.html?APNo=2272 (2018). (Accessed: 15 March 2019).

[9] NATO Standardization Office (NSO), "NATO STANDARD ISR STREAMING SERVICES-AEDP-18." https://nso.nato.int/nso/nsdd/apdetails.html?APNo=2273 (2018). (Accessed: 15 March 2019).

[10] NATO Standardization Office (NSO), "NATO STANDARD ISR WORKFLOW ARCHITECTURE-AEDP-19." https://nso.nato.int/nso/nsdd/apdetails.html?APNo=2274 (2018). (Accessed: 15 March 2019).

[11] "CORBA - Common Object Request Broker Architecture." http://www.corba.org/. (Accessed: 13 March 2019).

[12] Andersen, L., "Open JDK." JEP 320: Remove the Java EE and CORBA Modules https://openjdk.java.net/jeps/320. (Accessed: 12 March 2019).

[13] Bernstein, P. A., Bernstein, P. A., and Goodman, N., "Concurrency control in distributed database systems," *ACM Comput. Surv.* **13**, 185–221 (June 1981).

[14] Sipser, M., [*Introduction To The Theory Of Computation, International Edition 2E*], Cengage (2006).

[15] Eastlake, D. and Hansen, T., "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)," RFC 6234, RFC Editor (May 2011). http://www.rfc-editor.org/rfc/rfc6234.txt.

[16] Carol Peters, C. T., [*Research and Advanced Technology for Digital Libraries*], Springer Berlin Heidelberg (1997).