Secure Multicast

Jack Visoky
Security Architect and Sr. Project Engineer
Rockwell Automation

Joakim Wiberg
Team Manager Technology and Platforms
HMS Industrial Networks

Presented at the ODVA
2018 Industry Conference & 18th Annual Meeting
October 10, 2018
Stone Mountain, Georgia, USA

Abstract

Most security protocols, including TLS, are point-to-point solutions that do not support multicast. However, multicast can be extremely useful, especially for I/O in an industrial environment. Despite this, no single standard has emerged as the de facto method for multicast security. Rather, there are several potential methods for securing multicast traffic, each with advantages and disadvantages. These methods, and the potential applicability to multicast CIP traffic, are explored in this paper.

Keywords

Multicast, Secure Multicast, TLS, DTLS, IPsec, GDOI

Definition of terms

Asymmetric Cryptography	A cryptographic scheme which relies on different parties having different key material. This is generally used for signing and verification of data, as well as authentication and key negotiation. Asymmetric cryptography is generally significantly slower than symmetric cryptography. See also: Symmetric Cryptography.
DTLS	Datagram Transport Layer Security: a version of TLS that does not rely on guaranteed message delivery. This protocol is very similar to TLS with a few exceptions to allow for out of order transmission and non-reliable packet delivery (such as a sliding window of acceptable sequence counts on packets). See also: TLS
HMAC	Hashed Message Authentication Code: a MAC that is generated using a cryptographic hash. See also: MAC.
IETF	Internet Engineering Task Force: the most widely recognized, participated in, and used Internet standards body which develops open standards through open processes.

IPsec	A commonly used security protocol that implements security at the IP layer. Although many of the same guarantees are provided by IPsec as TLS, IPsec works at a lower layer in the protocol stack. See also: TLS
MAC	Message Authentication Code: a piece of data that can be used to verify the authenticity of a message. MACs are used extensively in secure communications to ensure the cryptographic authenticity of the data being transmitted.
RFC	Request For Comment: the de-facto Internet standards documents produced and managed by the IETF. Not all of these documents serve as normative standards, but many of the technologies that define how the Internet works are specified through IETF RFCs. See also: IETF.
Symmetric Cryptography	A cryptographic scheme which relies on different parties having the same key material. This is generally used for encryption/decryption of data as well as data authenticity. Symmetric cryptography is generally significantly faster than asymmetric cryptography, but has the downside that all parties need the same secret key. See also: Asymmetric Cryptography.
TLS	Transport Layer Security: the most ubiquitous secure communications protocol in use today. This protocol is defined by the IETF in a series of RFCs. Although TLS 1.2 is the most widely used, TLS 1.3 has recently been published and is beginning to gain traction and adoption. See also: IETF, RFC.

Why multicast?

In general multicast is a term that denotes sending information to a group of nodes at the same time. The sender only sends one single copy of the message and yet the networking infrastructure ensures that all receivers receive the message. The sender and the receivers may or may not be on the same network; whether subnet routing is possible depends on the protocol used for sending the multicast messages. This is an efficient way to send data since the message is only sent once on each link in the network. Copies of the message are only created when the link splits.

In order to realize multicast, one must make different assumptions based on whether the messages are sent on the Internet or on a local network. On the local network with the same addressing space, the switches have to support distributing multicast packets. On the Internet where the receivers can be distributed all over the world and are located in different address domains, the routing protocol has to support multicast.

Multicast can be either one-to-many, as shown in Figure 1, or many-to-many. The most common case is one-to-many, which is utilized for I/O communication on EtherNet/IP. On EtherNet/IP, an originator can open a Class 1 connection to many targets, and for each data production all the consumers receive the same data. Possible use-cases for this could be to synchronize several motor controllers setting the same speed at the same time or several outputs being energized at the same time in an I/O module. The reverse option is also a common use-case, in which one originator opens a Class 1 connection that additional originators subsequently reuse and listen to. In this case it could be a photo eye sending its status or a barcode-reader sending the scanned barcode to many recipients.

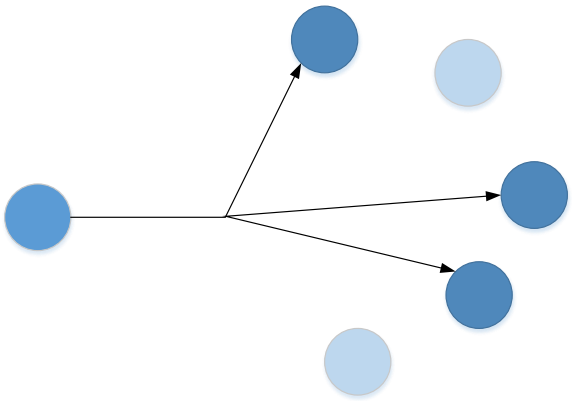


Figure 1 one-to-many multicast

EtherNet/IP multicast is accomplished using IP multicast and layer 2 delivery on Ethernet. By opening an IP multicast connection on EtherNet/IP, an IP number from the IP multicast group address is selected. This is done based on specific rules outlined in Volume 2 of the CIP Networks Library. The IP multicast packets are delivered using Ethernet, and a dedicated Ethernet MAC address range is reserved for this purpose. The IP multicast address is mapped to an Ethernet MAC address space.

IP Multicast Security Challenges

Unicast communications by definition are point-to-point: the sender addresses one known intended receiver with a unique address. In the case of multicast, the sender in many cases doesn't know who is receiving the messages. Messages are just sent to a group address and picked up by the members in the group. This creates some unique and complex security related issues.

One important factor to consider with secure multicast is the group size, i.e. the number of receivers. Based on the protocol that is used to secure the multicast traffic, the number of members in the multicast group might impact the performance. The efficiency of the multicast communications does not necessarily scale well with the group size, as the group communications may be efficient for a small amount of members in the group but not when this grows to a large group. As the number of members increases, the group begins to see performance issues. Furthermore, a larger secure multicast group where the frequency of members entering and leaving the group is high might also cause a problem because of performance required to authenticate new members. This can be especially problematic as these situations can come in bursts and are difficult to predict, leading to variability of performance.

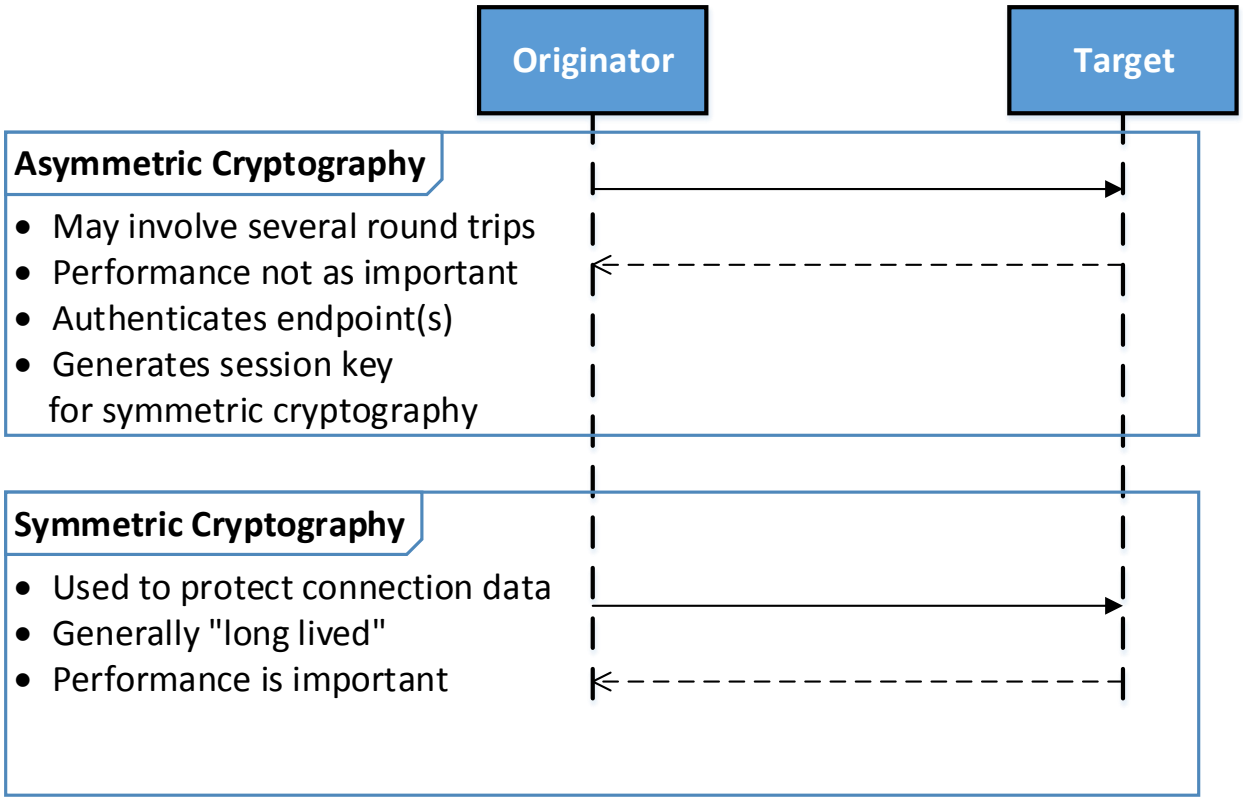
The scalability of secure multicast is particularly important when it comes to extending the security features for a larger group of senders and receivers and possibly also distributed over a larger physical area. These conditions need to be handled with minimal performance degradation for a system as a whole. Specifically for the secure multicast case, scalability relates to the management and distribution of cryptographic keys and other security related credentials and policies.

Naturally the management of the cryptographic keys and credentials needs to be protected, thus the issue of trust is important. There must be an underlying model of trust for the secure multicast communication. This model of trust must be implemented by all entities that are a part of the secure multicast communication.

Cryptographic Challenges in Secure Multicast

There are a few cryptographic properties that are used in most modern secure communications that are particularly difficult for a multicast scheme to incorporate. Central to this is the fact that symmetric cryptography is significantly more performant than asymmetric cryptography; that performance benefit is generally leveraged by secure communications protocols. However, as its name would reveal, symmetric cryptography necessitates all parties having the same key, which allows for all parties to be able to (cryptographically) perform the same operations. This has implications for multicast communications.

The basic structure of secure communications protocols is to use asymmetric cryptography for authentication and key agreement, and then use symmetric cryptography for data encryption and data authenticity. This is a paradigm that is seen throughout the point-to-point secure communications protocols (TLS, IPsec, etc…). The computationally intensive operations involved with asymmetric cryptography make it difficult to use outside of this initial connection setup, although for the connection setup it is invaluable as the two communicating parties usually do not have any shared secret information that they can use for key material. The diagram below gives a very brief description of the two stages generally involved in a secure communications protocol:



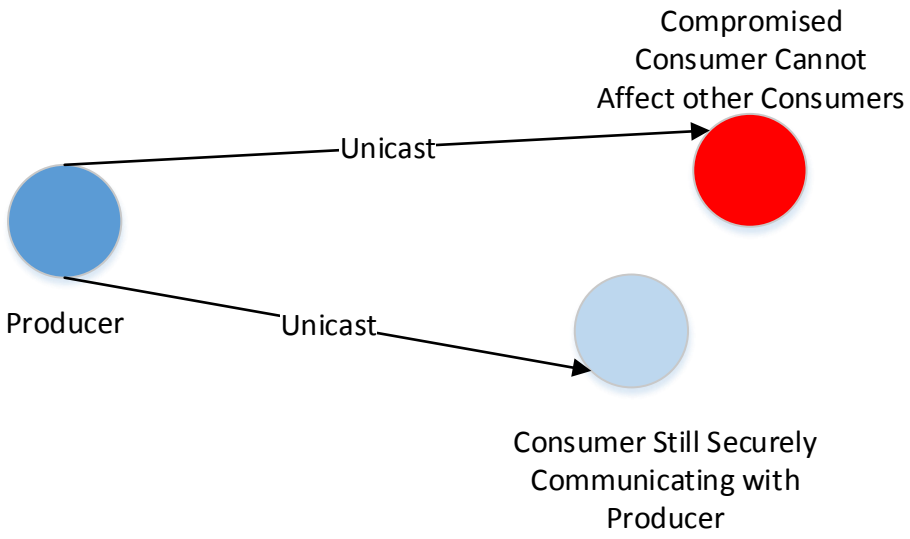
Mapping this type of scheme onto a secure multicast protocol generally produces some issues that have to do with the nature of the symmetric cryptography. Although potential mitigations exist (and are explored in this paper) there is no obvious and robust solution that provides the same benefits and assurances as in the unicast security protocol case. These issues are explored below

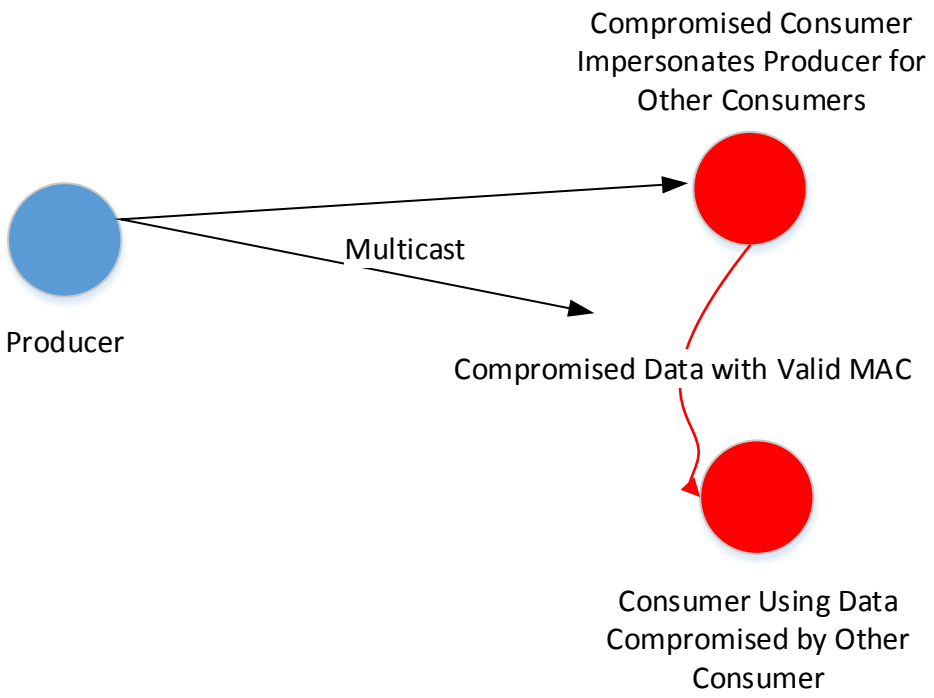
Issue #1: Lack of Originator Authenticity

Within a unicast security protocol, the data is almost always sent with some authenticity assurances. This often comes in the form of an HMAC, although the popularity of authenticated encryption schemes is growing. However, these schemes all use symmetric cryptography for data authenticity. In a unicast case with only one target and one originator, the use of symmetric cryptography provides very robust assurances. This works as follows: the originator and the target both have the same key that can be used to generate a MAC on the data. Any outside attacker does not have access to this key and therefore is not able to modify data in transit or produce original data on the connection. However, because there is only one originator and one target, this scheme gains an additional property. For either entity, there is an assurance of which entity produced the data. For example, if the target receives a packet with a valid MAC, it knows with certainty that the originator produced this packet. This is trivial because there is an assumption that only the originator and target have the session key, and if the target did not produce the packet then the originator must have produced it. The same can be said of the

originator when receiving responses from the target. Note that this case easily generalizes to a scheme that is less client-oriented and more peer-to-peer, as long as there are only two parties participating in the secure communications.

Contrast this mechanism with the multicast use case. A simple multicast case would have one data producer and multiple data consumers. As an example take the most trivial of these cases: one data producer and two data consumers, the producer is sending multicast data to each consumer. Once again, assume the use of symmetric cryptography to generate a MAC on the data. In this case the key must be shared securely between the three endpoints, which can be assumed to have occurred during the handshakes. In this case the three endpoints can still have the same assurance that an outside attacker is not able to modify the data in transit or produce original data due to the fact that the MAC key is only known by these three endpoints. However, there is an assurance that was present in the unicast case that is lost in the multicast case. In the unicast case there were only two parties, as such it was trivial for either party to know who produced a given packet (put another way, an endpoint knows all the data it produces, any other data with a valid MAC is necessarily produced by the other endpoint). However, in this case, any data consumer has the ability to impersonate the publisher or to impersonate any consumer. In some applications this risk may be acceptable, however one could easily imagine scenarios where it is not. For example, there might be a system where a data producer is multicasting information regarding equipment temperature. There are data consumers that are using this information to make decisions regarding valves that will regulate system temperature, as well as business systems consuming this data for offline analytics. In this case, a compromised business system could affect the entire system by impersonating the data producer and causing the valves to misbehave, potentially placing the system in a dangerous state. The more consumers in a system, the more the possibility that one can become compromised and impersonate the producer. The diagrams below contrast the assurances in a unicast scenario versus this multicast scenario.



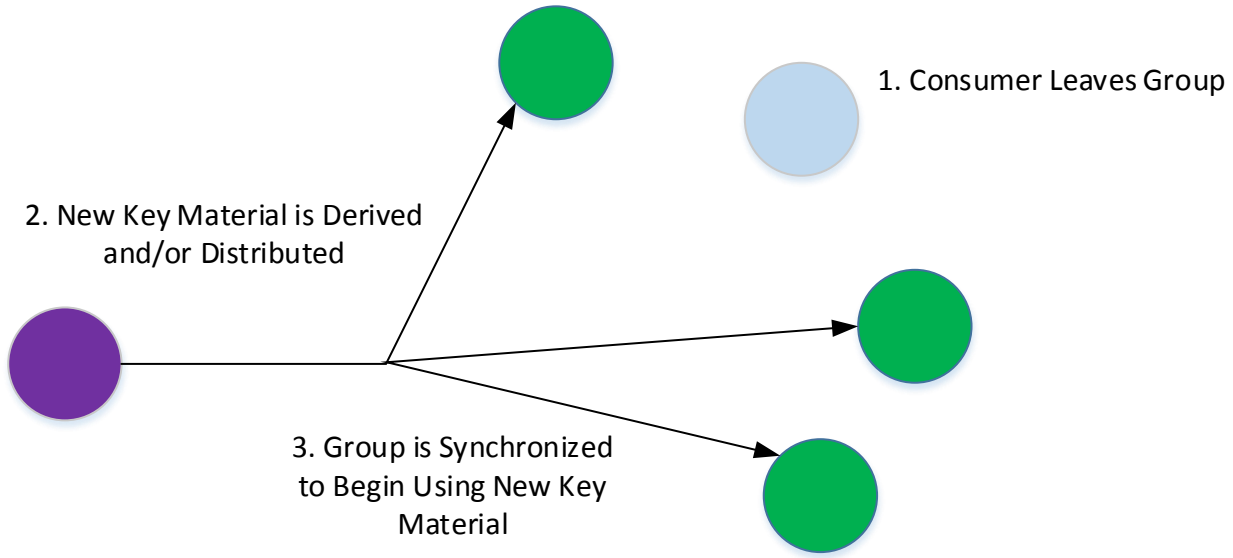


Note a potential solution to this would be for the producer to use asymmetric cryptography to sign the messages sent to the consumers. This would provide very robust assurances in that none of the consumers would be able to impersonate the producer or modify data. However, as previously mentioned asymmetric cryptography is very computationally intensive, far more than symmetric cryptography. In the vast majority of use cases it would not be feasible for messages to be signed and verified using asymmetric cryptography due to performance constraints. Another mechanism might be to use the network infrastructure to enforce communications rules. For the case discussed here, the infrastructure could enforce that the two consumers cannot communicate with one another. This may be a mitigation in some cases, although it could also introduce unwanted complexity and limitations into the network, besides requiring special support from the network infrastructure.

Issue #2: Consumers Dynamically Joining and Leaving

Another issue that is unique to multicast is the ability of data consumers to dynamically join and leave the multicast group. For the unicast case, this is again trivial. If an originator wishes to communicate with a target then a connection is set up which includes all of the necessary security operations (authentication, data encryption, etc…). Either the originator or target can close the connection at any time, in which case resources are released and no further action is taken on the connection. However, these operations become more complex in the multicast case. In the case where a new consumer wishes to join the multicast group the operations are likely similar to what occurs when setting up a unicast security connection. That is, there is likely a step with some asymmetric cryptography as well as a step where key material is derived and/or shared. At this point the new consumer has the appropriate key material to consume the data within the multicast group. However, within a multicast group there is also a possibility of consumers leaving the group. In some cases this might be treated as a very straightforward disconnecting from the group; the consumer simply closes its connection to the multicast group and the resources are released. However, this creates the possibility that key material for the current multicast group is still present on the recently discharged consumer. Given that this consumer was indeed at one time a legitimate member of the multicast group this may be an acceptable risk. However, it might also be the case that the risk of this key material continuing to exist on an endpoint that is no longer within the multicast group is not acceptable. If that is the case then new key material must be generated/distributed to the remaining multicast members. Depending on the size of the multicast group, this could be a very

computationally expensive activity, especially given that it will be occurring at runtime rather than connection establishment time. Furthermore, this introduces challenges around synchronization: once all members have the key material there needs to be a point at which the produced data is switched to being protected with the new key material. For groups which have consumers often joining and leaving, this operation could eventually scale to unacceptable limits where the system cannot meet performance requirements.



Of course, this is only an issue if there is shared symmetric key material within the multicast group. However, not using this type of symmetric cryptography will introduce major performance burdens on every message produced and consumed within the group, as previously discussed. In only the most extreme scenarios (messages sent extremely infrequently) would this be likely to be acceptable. That is, for I/O communications, using asymmetric cryptography for message protection is simply not an option.

This discussion focused on the case of a consumer leaving the group and the possibility for it to access or even modify multicast data. Note that there is a parallel case with a new member joining the group. In this case there might be a security requirement that the new member is not able to access old multicast data that was produced prior to this member being an authenticated part of the group. Had that member been on the network and capturing packets it might be able to decrypt these packets if the key material is the same. As such this is a very similar security concern to the one just discussed with members leaving the multicast group. However, various use cases might be concerned about one, both, or none of these scenarios. A threat model should be used to drive the decision about whether or not these scenarios are of concern for a given application. However, a general solution would provide mechanisms for mitigating all of these risks, whether or not the mitigations are indeed important in all cases.

These challenges are not insurmountable, and mitigations can be implemented. However, both of these challenges stem from the fundamental nature of symmetric cryptography, and as such they are extremely challenging to mitigate in a way that is workable in a general case. The multicast technologies discussed in this paper each deal with these issues in various ways, as will be discussed further. However, it is important to note the limitations of cryptographic primitives, especially symmetric cryptography, that are in common use today and how that affects any potential multicast solution.

Existing IP Multicast protocols

This section gives a brief description of some options for secure multicast communication, ranging from commonly used implementations to suggested additions to protocols not supporting secure multicast today.

IPsec and Group Domain of Interpretation

IPsec is a protocol suite that implements security, authentication, confidentiality, and integrity, for application and data over IP. IPsec uses three mechanisms to implement and realize the secure communication on IP:

- Authentication Header
- Encapsulation Security Payload
- Security Association

The authentication header and the encapsulation security payload provide authentication, confidentiality, and integrity functionality to the IP communication. Before secure communication can be carried out using the authentication header and the encapsulation security there must first be a security association. The details for how this security association is set up are very complicated, though it all comes down to setting up shared security attributes such as cipher, key, and hashed message authentication.

The shared security attributes are set up between two hosts before any data transfer takes place. The cryptographic keys can be of either a shared secret or X.509 digital certificates obtained from a certificate authority. There are several techniques and technologies used to automate the deployment of the digital certificates in the hosts. One of the more common protocols used when setting up the security attributes is Internet Key Exchange (IKE).

Both IPsec and IKE are designed to work as point-to-point protocols securing the data communications between two endpoints or two routers, allowing all hosts on one side of the routes to securely communicate over the internet with all hosts on the other network. In order to implement secure multicast using IPsec and IKE a protocol called Group Domain of Interpretation (GDOI) may be used. GDOI is a group key protocol where group members register with a key server. The key server authenticates the group members and distributes the cryptographic keys and other security attributes.

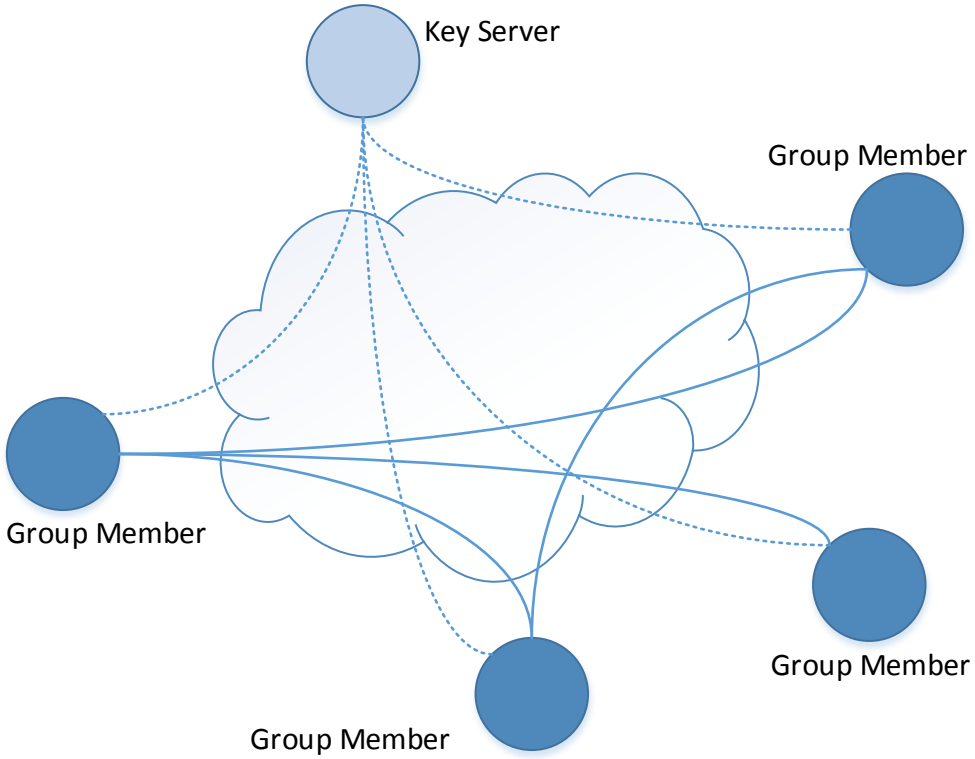


Figure 2 GDOI Example

Figure 2 illustrates an example with a Key Server and four Group Members. In the example the four Group Members register with the Key Server, which authenticates them, and downloads the security attributes needed for IPsec. Once a Group member is registered and authenticated it has the keys

necessary to encrypt and decrypt packets sent to and received from other group members. At this point the Group Member can exchange IP Multicast packets, using IPsec, with all other Group Members who have been authenticated by the Key Server.

Besides authenticating the Group Members, the Key Server is responsible for maintaining the keys for the group. Rekeying takes place on a regular basis depending on the security policies around the renewal of the key. When the keys are about to expire, the Key Server will push out new keys to all Group Members forcing them to update the keys used for encryption. The Key Server has the ability to push out policy to the group members that denote key expiration, key refresh, and other such properties. Note that the members have the ability for both pull and push of new keys and policies, although these operations are fairly complex and generally require several round trips from group members to the Key Server.

The ability for the Key Server to re-key the members is the mechanism by which GDOI deals with the issue discussed previously, namely group members leaving the group. This re-keying prevents the key material from being present on members that have left the group, although there is often a window of time for which the key material is still valid even after the member has left. Depending on the risk tolerance for this, the policy could be set such that re-keying happens often, although of course that would be at the expense of performance. Certainly this is a method for dealing with members leaving the group, although the performance trade-offs must be considered when evaluation a policy around this.

Note that GDOI and IPsec do not deal with the first issue of multicast security that was discussed, namely that the key material is shared amongst the members. GDOI does allow for policy around which members may send or receive data, although this is not enforced through cryptography, but rather by endpoint policy (the GDOI RFC states this explicitly: “GDOI ultimately establishes keys among members of a group, which MUST be trusted to use those keys in an authorized manner according to group policy.”). As such, GDOI and IPsec do not provide strong assurances around the use or misuse of group keys by members within the group. This may be perfectly acceptable in some cases but not necessarily in all cases.

There are a lot of options within GDOI, and although this brings a good deal of flexibility, it also creates a lot of complexity. Furthermore, the only protocol which really works with GDOI in practice is IKE and IPsec. As CIP Security relies on TLS and DTLS, there are completely different key agreement mechanisms within these protocols compared to IKE; therefore, it would not be straightforward to use GDOI and IKE within CIP Security. A quote from the GDOI RFC shows the lack of support for other protocols (emphasis added) “Besides ESP and AH, GDOI should serve to establish SAs for secure groups needed by other Security Protocols that operate at the **transport**, application, and internetwork layers. These other Security Protocols, however, are in the process of being developed or do not yet exist.” However, no support for TLS within GDOI has been forthcoming. In fact, although GDOI is actively used, it is somewhat of an older protocol; note that the RFC states that nodes should support SHA-1 signatures (which have been shown to be breakable and are no longer recommended for use by the general cybersecurity community). For these reasons it appears that no work is being done to support TLS. Also, the presence of a Key Server itself could be somewhat concerning as this would be seen as a single point of failure for the operation of each Group Member. Secure multicast solutions often have a Key Server, although this is one of the concerns that would need to be mitigated within whatever technology is used to implement secure multicast in CIP Security.

Although GDOI is a standardized and deployed technology, its availability within open source or commercial libraries is fairly limited. GDOI is not widely supported, which would likely force device vendors to implement or pay a third party to implement this protocol. In addition to this, GDOI is fairly complex and would add quite a bit to the memory requirements of CIP Security endpoints.

Secure Multicast using DTLS

Transport Layer Security (TLS) is today the prevailing protocol used to realize secure communication of web traffic on the internet. Likewise TLS is also used to provide an end-to-end security communication session for other communication protocols, with CIP Security for EtherNet/IP being one of them. This end-to-end security session provides pairwise communication between two endpoints. TLS doesn’t deal with

loss of packet and/or packet reordering, thus it relies on TCP to provide the reliability in the communication. A growing number of applications are using UDP for communication. However TLS cannot be used on UDP as UDP doesn't provide any form of reliable communication regarding loss of packets and packet reordering. Instead Datagram Transport Layer Security (DTLS) has been introduced. DTLS adds modifications to TLS to allow it to communicate over UDP. After modifications DTLS still maintains the majority of functionality from TLS and preserves compatibility on the application layer. UDP and DTLS is also used to secure the IO communication with CIP Security for EtherNet/IP.

UDP is likely the most commonly used transport layer protocol for realizing multicast communication for IP traffic. With UDP there's no connection established between the endpoints that are communicating. This is in contrast to TCP which creates a session between the two parties. Since there's no session or connection established and the datagrams are just sent out on the wire, it is possible to send the data to many receivers. The UDP datagrams are sent out to dedicated IP addresses which belong to special IP multicast groups. The infrastructure learns, using IGMP, where the receivers are on the network and makes sure that the datagrams are delivered to the multicast receivers. Even if DTLS is using UDP as a transport layer it's not possible to send multicast data using DTLS. The primary reason is that during initial handshake, when setting up the DTLS session, the two parties, amongst other things, agree on cryptographic algorithms and exchange keys used for subsequent communication. The outcome of this initial handshake which is a part for the session negotiation process is a secure private session between the two parties.

Neither the current version of DTLS, version 1.2, nor the upcoming version 1.3 makes it possible to send secure multicast data. However with the increasing adaptation of Internet of Things this has been recognized as an important feature missing in DTLS. Efforts to add secure multicast support for DTLS have been initiated years ago within Internet Engineering Task Force (IETF). This effort stopped at the beginning of 2015, and since then no effort within the IETF has existed to pick up the activities. There has however been several other related work topics looking into adding support for secure multicast on DTLS. None of those efforts have been focused on bringing the work into IETF and making it an open standard.

There exists several projects that have looked into adding support for secure multicast to DTLS. In most cases they build on and make use of existing technologies like TESLA and ECDSA signature for Source Authentication. Likewise, the projects have been using the standard Group Secure Association Key Management Protocol (GSAKMP) to manage the key distribution within the secure multicast group. Some of the projects have proven that it's possible to implement and make DTLS support Secure Multicast using already existing standards with minimal changes to the DTLS protocol. Many of these projects have the concept of a secure key server embedded within them, thus the same concerns about the failure of this server affecting system operations still exist. Furthermore, many of these projects leave some details of how a practical multicast security system would work open-ended. This would have major concerns for both the implementation of a system, as well as the interoperability of endpoints.

Although there is no official DTLS multicast solution, the RFC documents suggest that a similar tactic was taken to that of GDOI for the two multicast security issues identified earlier in this paper. That is, policy can control when the group is rekeyed, which prevents members from accessing data outside of their time within the group (to a point, as rekeying will likely not occur atomically with a member joining or leaving). The issue of shared symmetric keys is also not dealt with in the proposed DTLS multicast solutions, although again note that these are incomplete. If the effort to develop DTLS multicast is renewed within IETF then this status could certainly change.

OPC-UA Pub Sub Security

One solution specifically tailored to the industrial communications space is that of OPC-UA Pub Sub. Pub Sub is a relatively new mechanism that allows for multicast of data. This mechanism can be done through a broker (e.g. MQTT) or without a broker. Pub Sub security relies on the presence of a Secure Key Server (SKS) to provision the publishers and subscribers with keys. The SKS can change keys according to the configured policy, which if configured correctly can mitigate the concern around subscribers accessing historical or future data. Note that unlike the IETF published solutions, OPC-UA is

a closed solution which is available only to members of the OPC Foundation. As such the public details available are limited and therefore a full evaluation of the solution is not possible in this venue.

Cellular Network Secure Multicast

One place where multicast is still used is within cellular communication networks. At the heart of the modern cellular networks is 3GPP technology; all modern cellular communication networks are built upon this technology. Many cellular communications technologies (3g, LTE, etc…) have an option for multicast with security. Security for 3GPP is documented in [7]. Fundamentally, this multicast security solution still has the notion of a secure key server, which can again allow for rekeying and key expiration to limit access to historical and future data. Furthermore, as cellular communication is quite different from EtherNet/IP, there are a number of technologies in use that would not easily fit in with existing EtherNet/IP architectures, including SRTP for secure transport and HTTP digest authentication for endpoint authentication. Although there are certainly many interesting ideas and technologies at work within cellular networks for both security and multicast communications, it would be quite difficult to apply the work done here to CIP Security in a way that did not fundamentally change EtherNet/IP. As such, these technologies would only serve as "inspiration" for a CIP Security multicast solution, as it would likely be difficult and expensive to build and maintain any solution based on this technology.

Unicast Only Option

The discussion of existing secure multicast technologies shows not only a lack of a clear "winner" in this space, but even a lack of energy around development and support of secure multicast technologies. It is instructive to consider the reasons for this, as this will potentially drive a course of action within CIP Security. Although multicast still has some very important uses, it appears to have fallen out of favor in many cases. There just aren't an incredible amount of uses of multicast communications within general Internet communications. It may even be illuminating to draw an analogy from Internet communications several years in the past. Previously, it was considered quite a burden to use security on web communications, hence most websites were simply using HTTP without any transport security. However, currently the vast majority of web traffic is secured with TLS, the same point-to-point security technology used in CIP Security. Improvements in processing power and network efficiency have made this possible (according to Firefox telemetry data, around 75% [6]). Even multimedia traffic is now often secured with TLS; as an example Netflix delivers all of its streaming content to customers secured via TLS [5]. Although this doesn't invalidate the usefulness of multicast and/or a secure multicast solution, the lack of energy around this coupled with the overwhelming adoption of TLS [4] point to a general trend of replacing multicast traffic with point-to-point unicast and associated unicast security like TLS and DTLS.

Suggestions supporting Secure Multicast in EtherNet/IP

As stated previously non-secure EtherNet/IP communication relies heavily on multicast. It would be a great benefit if CIP Security for EtherNet/IP would also provide options to support multicast communication. CIP Security for EtherNet/IP is built on top of TLS and DTLS. Since there is no support in DTLS for secure multicast, CIP Security subsequently had no support for secure multicast. However it was recognized that there may be a future need for a security solution to provide protection to multicast traffic.

Since CIP Security for EtherNet/IP is built on TLS and DTLS, it's natural to look into the option of using secure multicast on the security protocols already used. But as noted earlier there's no standard available to support secure multicast on DTLS. None of the technologies explored are simple drop in solutions; all would require significant development in order to be leveraged for protecting multicast EtherNet/IP traffic. Therefore, there are essentially two recommendations of this paper.

1. **Do Nothing**: If multicast EtherNet/IP is truly not used in most cases, and users are accepting of the multicast security limitation, then it is not worthwhile to invest the time and effort in creating and implementing a multicast security solution. As noted in the previous section on unicast, the market outside of industrial automation/communications seems quite lukewarm on the use of multicast. As such, there is a lack of a common solution. This could very well be a sign that the benefits of securing multicast are not worth the effort involved in implementing and securing it. If the assumption that the use of multicast is very uncommon, those users could focus on hardening

their network infrastructure to prevent network level attacks. Of course it is always better to have security directly supported, but network hardening might be sufficient if the use cases are limited.

2. **Develop Multicast DTLS within IETF**: If users truly demand a multicast security solution, then a proper effort to develop this technology should be undertaken within IETF. ODVA member companies could collaborate here with other technology leaders to develop a robust and purpose-built solution for DTLS multicast security. Many of the examples like GDOI and IPsec multicast security technologies could be used as inspiration for a DTLS multicast solution. However, being that DTLS is fundamentally different from IPsec, an actual effort to develop a new technology, rather than a simple re-use of something existing, would be most prudent and beneficial.

References

[1] ODVA, Inc. The CIP Networks Library, Volume 8: CIP Security™, PUB00299

[2] The Group Domain of Interpretation, RFC 6407, https://tools.ietf.org/html/rfc6407

[3] Datagram Transport Layer Security Version 1.2, RFC 6347, https://tools.ietf.org/html/rfc6347

[4] Let's Encrypt, https://letsencrypt.org/

[5] "It wasn't easy, but Netflix will soon use HTTPS to secure video streams" https://arstechnica.com/information-technology/2015/04/it-wasnt-easy-but-netflix-will-soon-use-https-to-secure-video-streams/

[6] Let's Encrypt statistics (derived from Firefox Telemetry Data) https://letsencrypt.org/stats/

[7] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security of Multimedia Broadcast/Multicast Service (MBMS) http://www.3gpp.org/ftp//Specs/archive/33_series/33.246

[8] OPC-UA – The Security Solution for the Internet of Things https://opcconnect.opcfoundation.org/2018/04/opc-ua-the-security-solution-for-the-internet-of-things/

[9] An Open Source implementation of OPC UA Publish/Subscribe over TSN and a related demonstrator to be exhibited at Embedded World 2018 in Nuremberg, Germany, Letter of Intent, https://www.osadl.org/uploads/media/OSADL-OPC_UA-TSN-LoI-V5.pdf

[10] The Multicast Security Toolkit, https://www.cisco.com/c/en/us/about/security-center/multicast-toolkit.html