AKM Core Technology Concepts

AKM Core Technology Concepts Overview

Autonomous Key Management is an open-source, fully decentralized, share-nothing symmetric encryption system, which has no cloud dependency, supports air gapped configurations, and is Quantum-safe. It is implemented as a drop-in replacement for PKI+TLS, and is a radically simplistic methodology for cyber protection of people, data, networks, resources, services, and devices that all but eliminates manual maintenance of the security network.

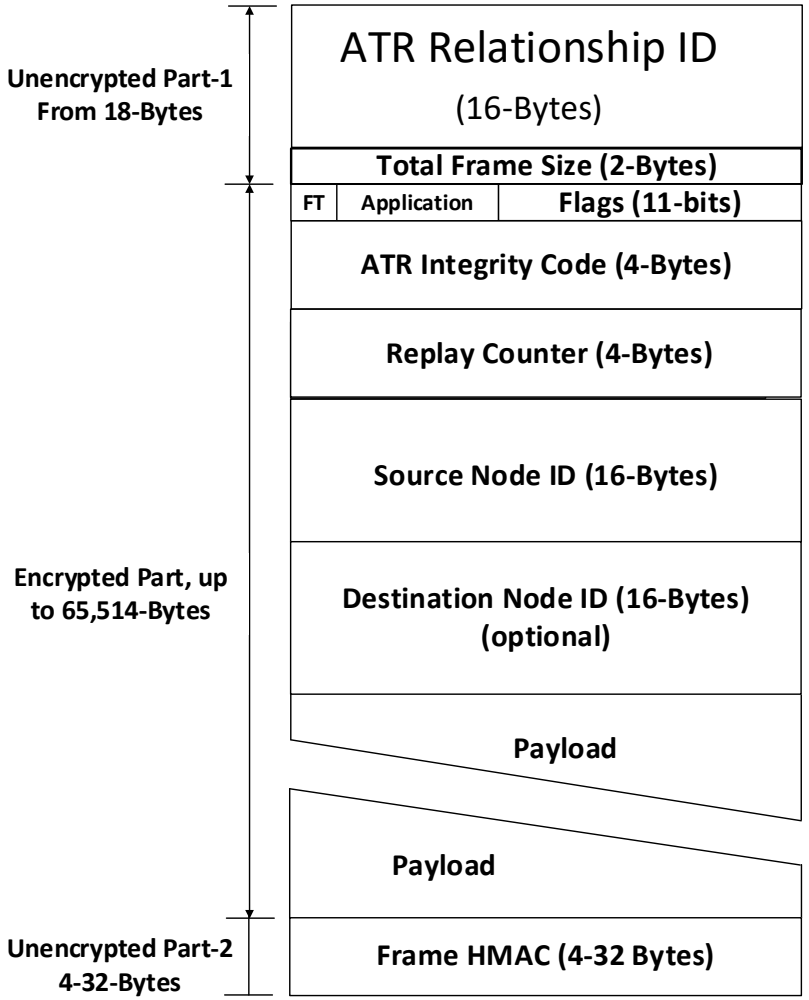
Given that **88% to 95% of all cyber-attacks are a result of human error**, managing passwords, certificates, keys, secrets or tokens, this alone provides a huge benefit in managing a cybersecurity platform, ideally, that implies that cybersecurity policy should be directly implemented and enforced without multiple layers or constant management. AKM achieves this by allowing security relationships to autonomously manage themselves. So, no more humans in the cyber protection loop!

Thus, once an AKM implementation has been initially configured, the network will manage itself (and recover by itself) if it needs to. Addressing all five pillars of what NIST states constitutes a complete cybersecurity solution:

1) Identify
2) Protect
3) Detect
4) Respond
5) Recover

Last, AKM also provides an Asset Integrity Management (AIM) add-on that adheres to CENELEC 50701 and ISA/IEC 62443, and is the perfect foundation in providing a customized and native SBOM feature into our system that autonomously monitors and runs for perpetuity with minimal overhead.

AKM Frame Structure & Protocol Highlights



* https://news.ycombinator.com/item?id=15950481

Unencrypted Part-1

RelationshipID – The Relationship Identifier is unencrypted and enables the Broadcast Architecture default behavior of AKM.

Total Frame Size – Size of the entire frame, including the Frame HMAC.

Encrypted Part

Frame Type Bit – This bit determines if the frame is a Data Frame, or a Command Frame (including Session Status reporting and/or queries, Integrity Management, or Data Analytics information).

Application – Optional field specifying a particular device application.

Flags – These bits indicate information about the frame, including whether or not there is an optional Destination Node Identifier field, and enable data frames to convey status and state information about the relationship, the node, and/or the frame and thus decrease the need for explicit command frames.

ATR Integrity Code (AIC) – The AIC is used to provide implicit zero trust functionality, given that every AKM Endpoint has previously authenticated both its endpoint and its membership within an AKM Trust Relationship.

Replay Counter – Replay Counter increments on every frame.

SourceNodeID – This identifies the sender of the frame.

DestinationNodeID – This optional field identifies the target recipient, if any, of the frame

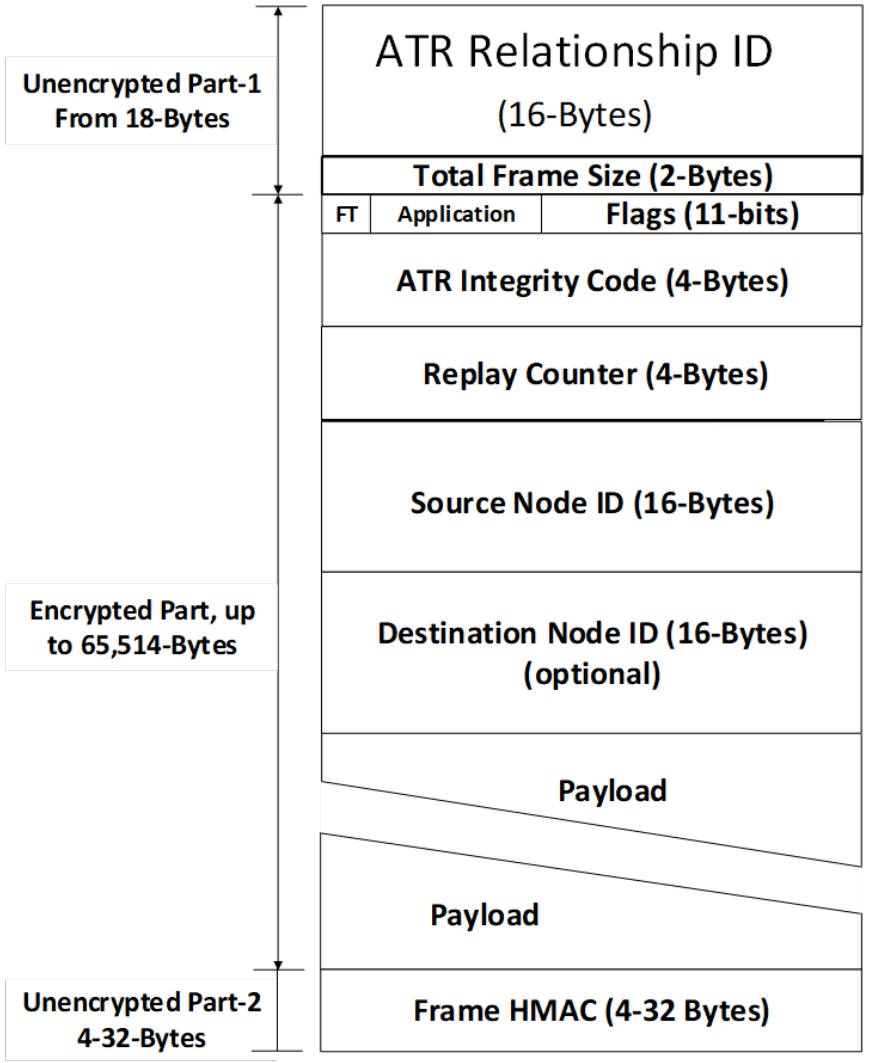
Payload – This is the encapsulated payload, representing the actual data being communicated.

Unencrypted Part-2

Frame HMAC – This is a SHA256 (or SHA512) based HMAC, that digitally authenticates the entire frame, including the unencrypted part of the frame header.

AKM uses a EtM scheme, meaning, it encrypts and then does the HMAC. As reported by Ycombinator Hacker News, "The recommended, more safer and sturdier method is EtM (Encrypt then MAC), in which you encrypt the plaintext then MAC the ciphertext. This way you can verify the cipher before you operate on it. EtM is [according to Wikipedia] "[...] the only method which can reach the highest definition of security in AE."

AKM Frame Structure & Protocol Highlights

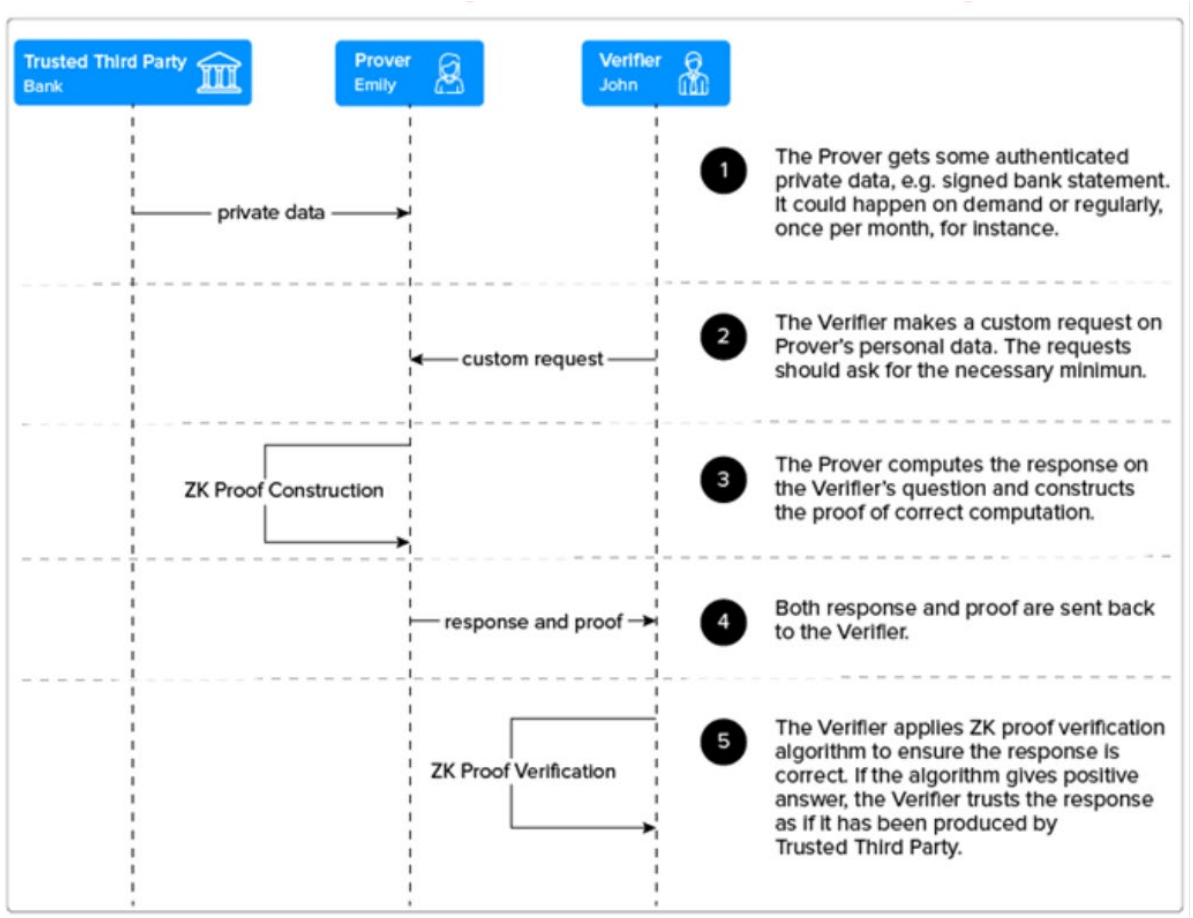


The AKM Transport Security Layer, together with the AKM KMS, are drop-in replacements for PKI & TLS/SSL (the TCP/IP stack security layers). Meaning, the AKM technology stack seamlessly integrates into existing technology communication and security standards and does not require further retrofitting.

Additionally, the AKM frame structure offers protection against MITM and Replay Attacks and Spoofing:

1) The presence of the "Replay Counter" ensures against repurposing previously sent frames and make it difficult to send modified imposter frames.
2) The "Frame Header HMAC Authentication" value authenticates the entire frame, including the unencrypted parts of the frame header. Thus, ensuring the frame header is not compromised.
3) The ATR Integrity Code authenticates the sender by ensuring the sender is part of the same ATR as the receiver and other nodes within the same ATR.
4) The "Replay Counter" is both encrypted and protected by the Frame Header Digital Signature, hence, preventing modification of it
5) There are no shared secrets that are ever exchanged or exposed that would allow an attacker insight into the AKM Security Credentials.

Zero-Knowledge Authentication

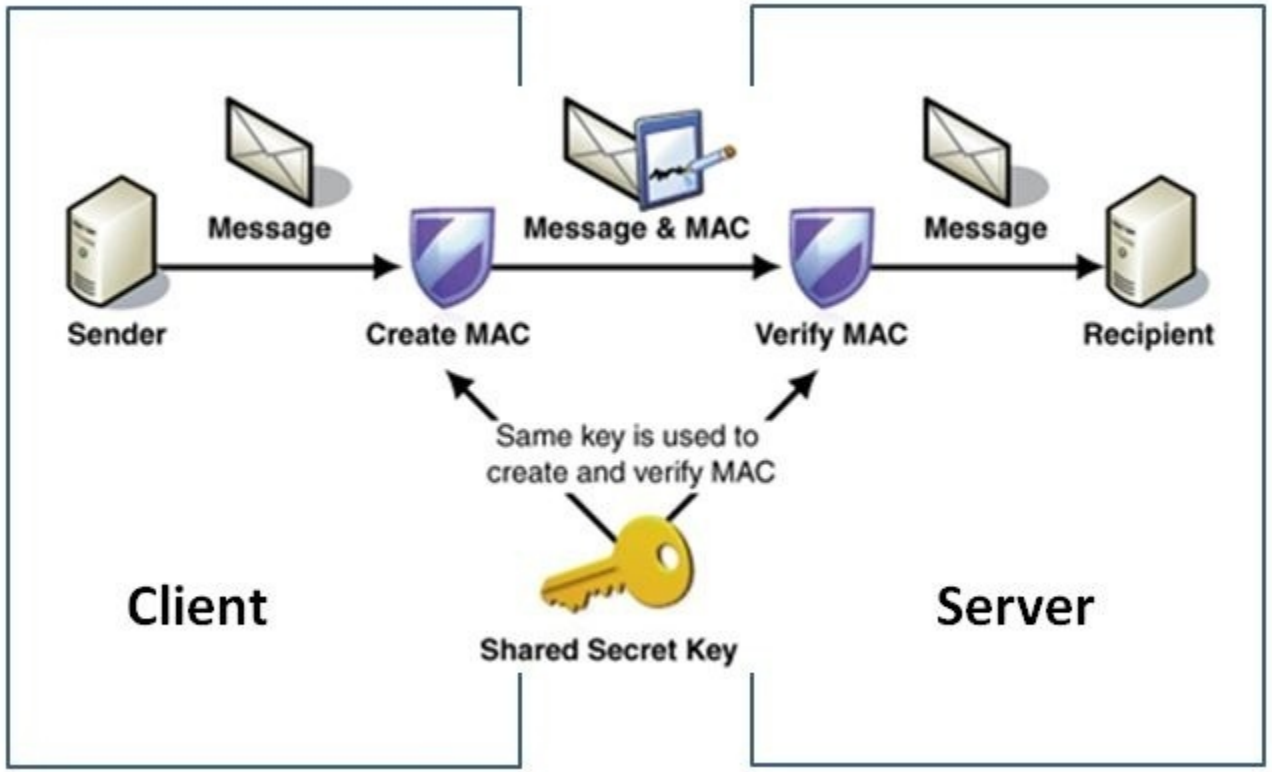


Zero Knowledge Protocol: Data Exchange

Applying ZKA roles within AKM:

- The *Authenticated Private Data* alluded to in Step-1 consists of two separate, unrelated sets of data. The first set, is the resultant Frame HMAC that is part of every data frame and is not accessible by the "Prover". The reason we can say it is NOT accessible by the "prover" is nuanced, in that the "prover" (i.e., the transmitting node) has no control over the value. The value is calculated based on parameters supplied by its own internal "verifier" (the secure enclave) and/or, ideally, if the Secure Enclave is of sufficient processing capacity, the *Authenticated Private Data* is calculated solely within the Secure Enclave at both ends. The secure enclave is "authenticated" by the ultimate Root-of-Trust (i.e., the Factory Backend Configuration [& Database] Server (FBCS)) at the time of initial provisioning and is hence a proxy of the FBCS, which is the ultimate Root-of-Trust. The second set of data that can be considered to be part of the "*Authenticated Private Data*", is the ATR Integrity Code (AIC) that is part of every frame and acts a token in a Zero-Trust Authentication framework.
- The "Verifier" is the Secure Enclave in the receiving node (which as mentioned above, is a proxy for the FBCS, the ultimate Root-of-Trust).
- The "Prover", which is transmitting node, supplies its calculated version of the "Private Data" to the receiving node. If the calculated version of the "Private Data" matches the "Authenticated" Private Data calculated by the "verifier" within the receiving node, then, the integrity of the ZKA relationship is said to have been authenticated.

Hash-Based Message Authentication Code (or HMAC)



HMAC Authentication is short for Hash-Based Message Authentication Code, a strategy used to simultaneously verify the integrity and authenticity of a message. This strategy is different from other authentication methods in the way that it used a cryptographic key along with a hash function. The algorithm behind the hashed message authentication code is complicated by hashing being performed twice. This helps in resisting forms of cryptographic analysis and protecting against threats. A hashed message authentication code is considered to be more secure than other similar message authentication codes, as the data transmitted and key used in the process are hashed separately.

Entropy of Security Credentials

The Age of the Universe as Measured in Seconds is between 10^{16} and 10^{17}.

nPr Function		
r-Subset Size	n-Set Size	P-Number of Permutations
15	128	1.72832E+31
20	128	2.91113E+41
25	128	3.89402E+51
15	256	8.75002E+35
20	256	6.82277E+47
25	256	4.78631E+59
15	512	3.54105E+40
20	512	1.05232E+54
25	512	2.97249E+67
25	1024	1.3466E+75

By using a relatively small vector of say, 128-bytes and then randomly selecting 15-bytes and then using those bytes as part of the process in creating security credentials, the number of permutations of even 15 different bytes from a single 128-byte vector is on the order of 10^{31}. Thus, making it impossible to brute force the resultant security credentials if nothing else is ever exposed.

The appeal of this methodology is that even if the 128-byte vector is published in clear text, the potential would-be hacker would still not be able to ascertain the correct credentials without other knowledge. So long as no secrets are exposed, even quantum computing will not be able to hack security credentials based on this method. Add to that, the fact that the vector is not published, is always encrypted when it is communicated, is periodically and randomly refreshed and that every security relationship has its own unique vector, this approach is easily categorized as quantum safe.

PKI Sharing the BEK vs. AKM sharing the PDV

One of the many advantages of AKM is that once configured, it never has to share any secrets in order to continue running.

AKM is said to be, "One and Done" with regard to its configuration and maintenance. Whereas, any time PKI wishes to update the Bulk Encryption Key (BEK), it has to potentially expose the BEK, which if intercepted, the system will then be considered to have been breached.

This is a major drawback of the entire PKI approach and despite many safeguards, has proven to be an Achilles Heel of the entire PKI framework. But, once someone understands AKM and finds out that the Parameter Data Vector (PDV) is also shared (and hence, potentially exposed), what is the difference between potentially exposing the PDV and potentially exposing the BEK?

ANSWER:

- If the BEK is exposed by a breached PKI framework, all data from that session is now also exposed.
- If the PDV is exposed by a breached AKM framework, nothing bad happens. There is no damage.

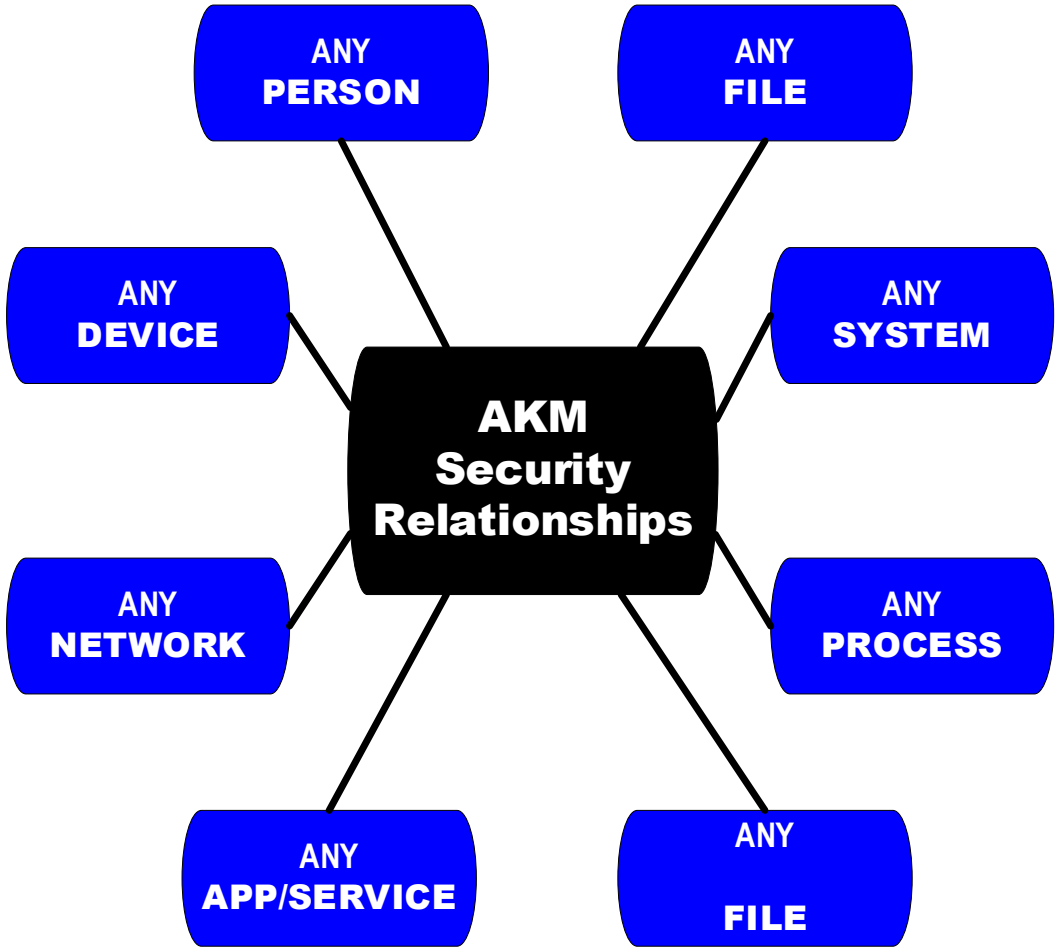
So, the next question is why doesn't exposing the PDV cause any issues for AKM?

ANSWER:

Unless the potential bad actor has access to the security credentials within the secure enclave itself (which explicitly by design, is not possible because the security credentials permanently reside within a separate tamper resistant hardware module and are NEVER exposed outside of that tamper resistant hardware module). Thus, there is nothing for the potential bad actor to base a potential breach on.

THERE IS NO STARTING POINT for the potential breach and this is also why AKM is said to be, "Quantum Safe"!

AKM Assets & the AKM Security Relationship (ASR)



The AKM provisioning process creates synchronized AKM Security Relationships (ASRs) using a multi-patented cryptographic virtualization model to structure AKM-enabled digital assets into functionally (contextually) defined security relationships.

Every AKM digital asset (representing both physical and virtual assets) is assigned a unique AKM identifier and can participate in unlimited concurrent security relationships. Each ASR has unique end-to-end security credentials, policy-based rules, and distributed ledger, and operates autonomously and 100% isolated from every other ASR, as well as the network hardware (and software) over which it transits, resulting in a virtualized multipoint-to-multipoint communications security mesh that is perfect for IOT and AI systems.

AKM Asset – An AKM Asset is the digital twin of a physical or virtual asset, which can be defined as a physical device, a person, an electronic image or file, or some organizational set of assets (ex., a group of AKM assets forming a subsystem).

AKM Security Relationships (ASRs) & the AKM KMS

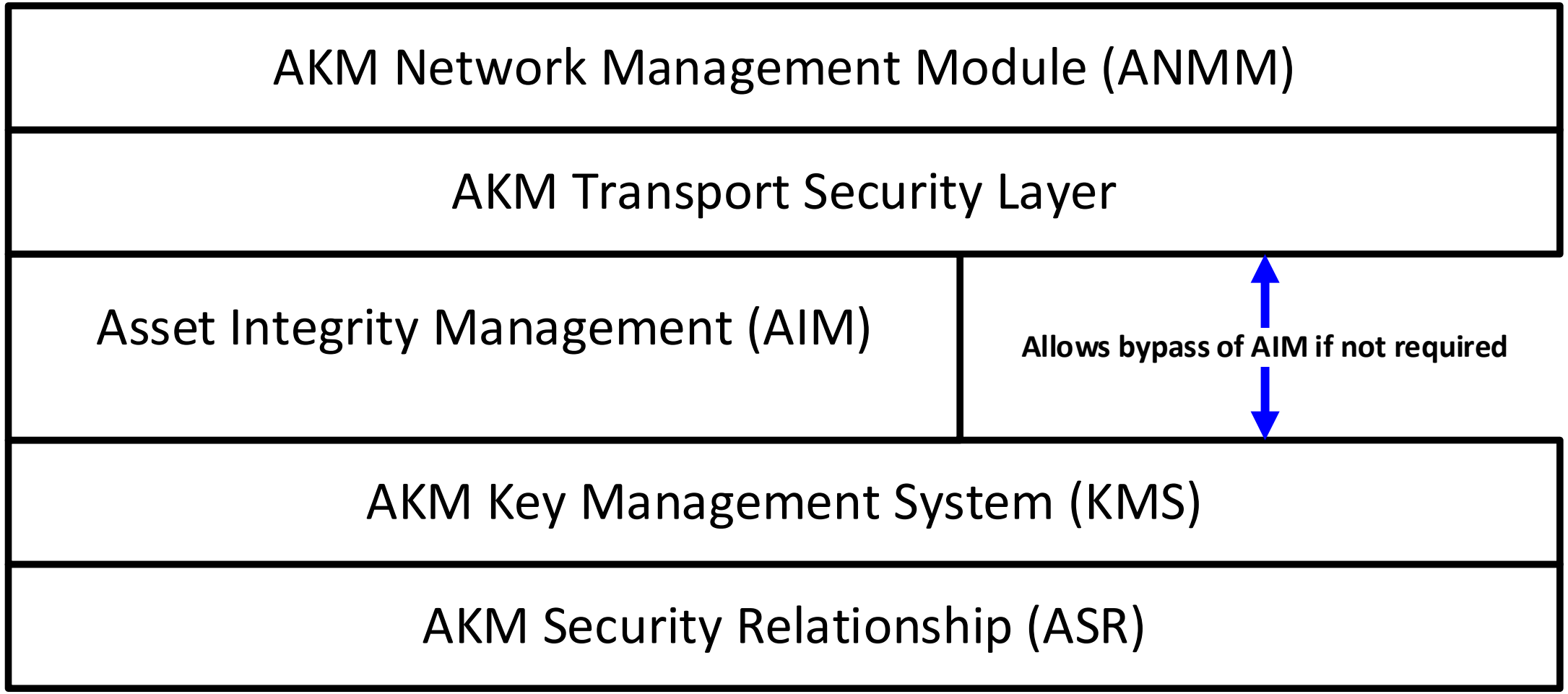
AKM is basically a technology stack (see the subsequent slide).

The first layer in the AKM technology stack is the concept of an AKM Security Relationship (ASR) that is the basis for the mechanics for how the AKM Key Management System (KMS) operates.

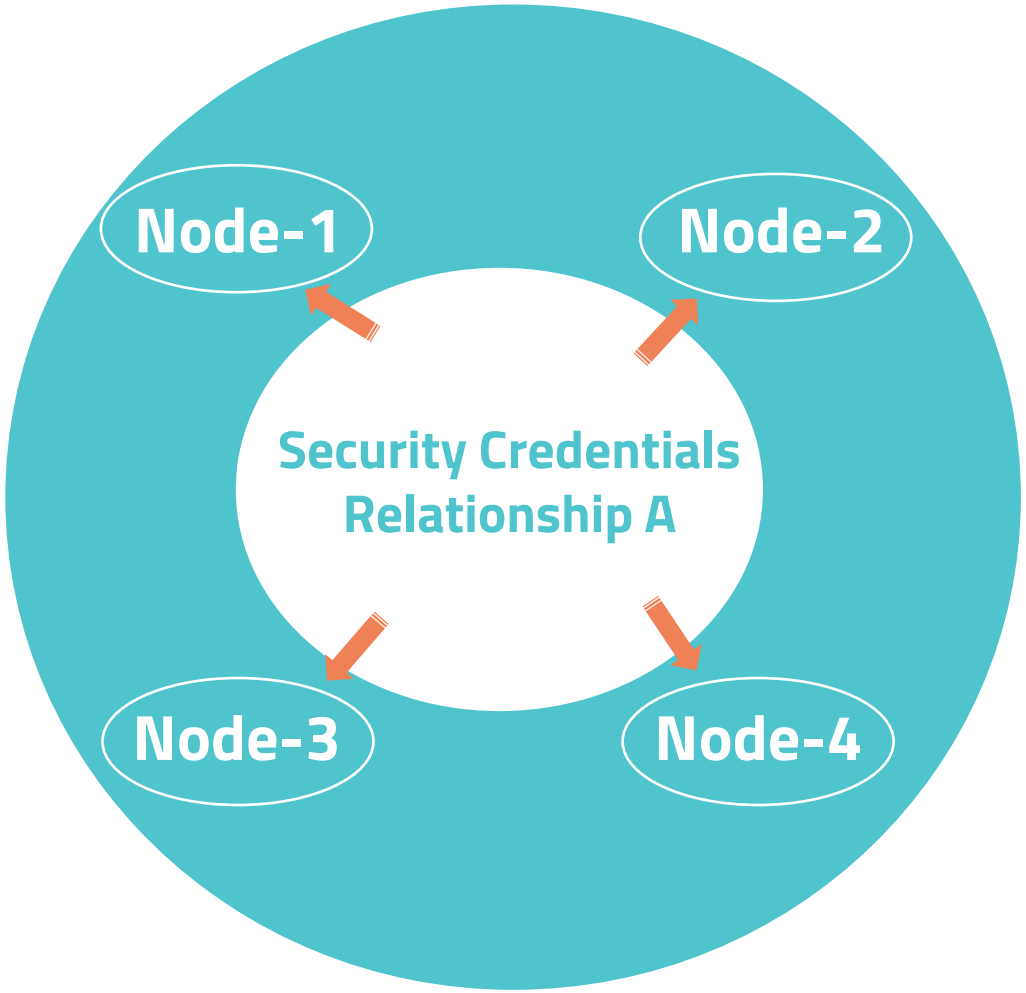
An ASR is nothing more than a group of two or more nodes that together form a secure cryptographic relationship. Key attributes of what constitutes an ASR includes, but is not limited to the following:

- All nodes within an ASR share a common set of security credentials.
- Security Credentials (which are maintained within a data structure called, the Synchronized Data Set (SDS)) are continually refreshed and do so in unison with the other nodes within the same ASR.
- Each ASR is unique and shares nothing in common with other ASRs, even other ASRs on the same nodes or set of nodes. This implies that ASRs provide virtual security groups, not physical security groups. Meaning, different processes on the same device may have different ASRs. This further implies that if somehow one ASR was breached, the breach would automatically be self-contained, because it would share nothing in common with the other ASRs.
- ASRs form end-to-end virtual security meshes, thus, enabling all nodes to communicate with each other via a virtual broadcast architecture.

AKM Technology Stack



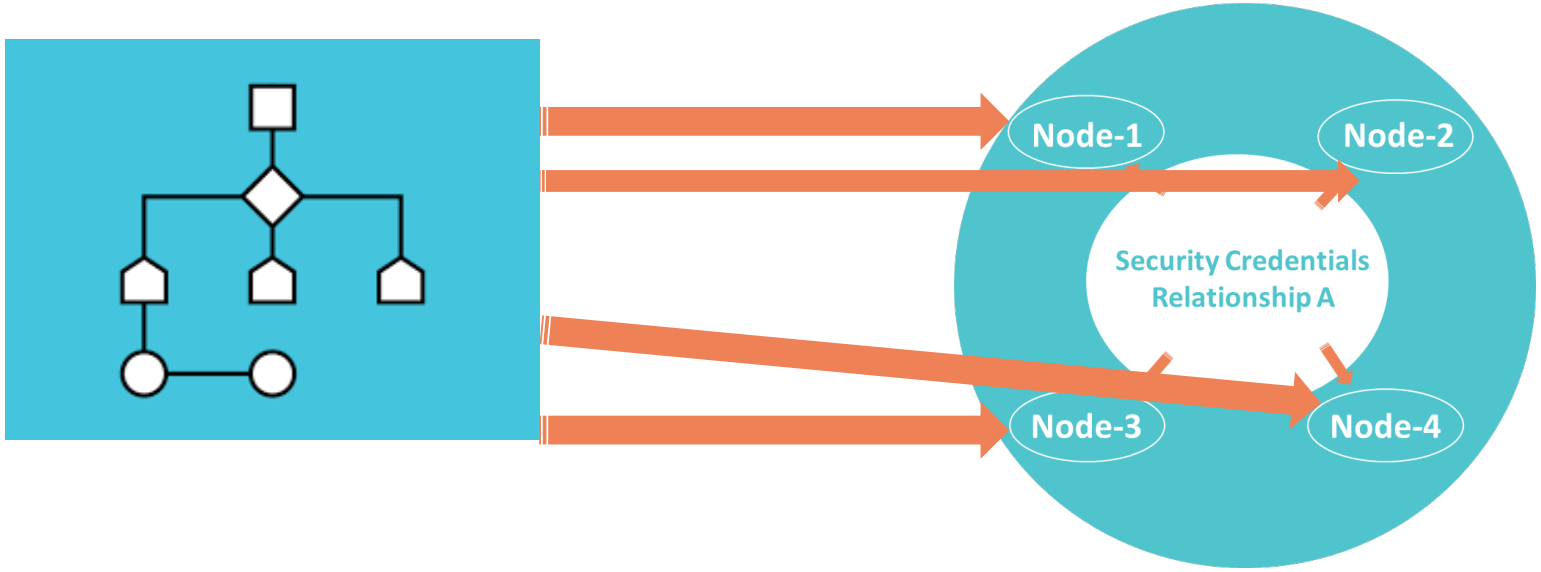
Multipoint End-to-End AKM Security Relationship (ASR) Diagram Example



AKM is able to provide true, multi-point, end-to-end secure communication, because it uses a decentralized, distributed architecture for maintaining and storing credentials

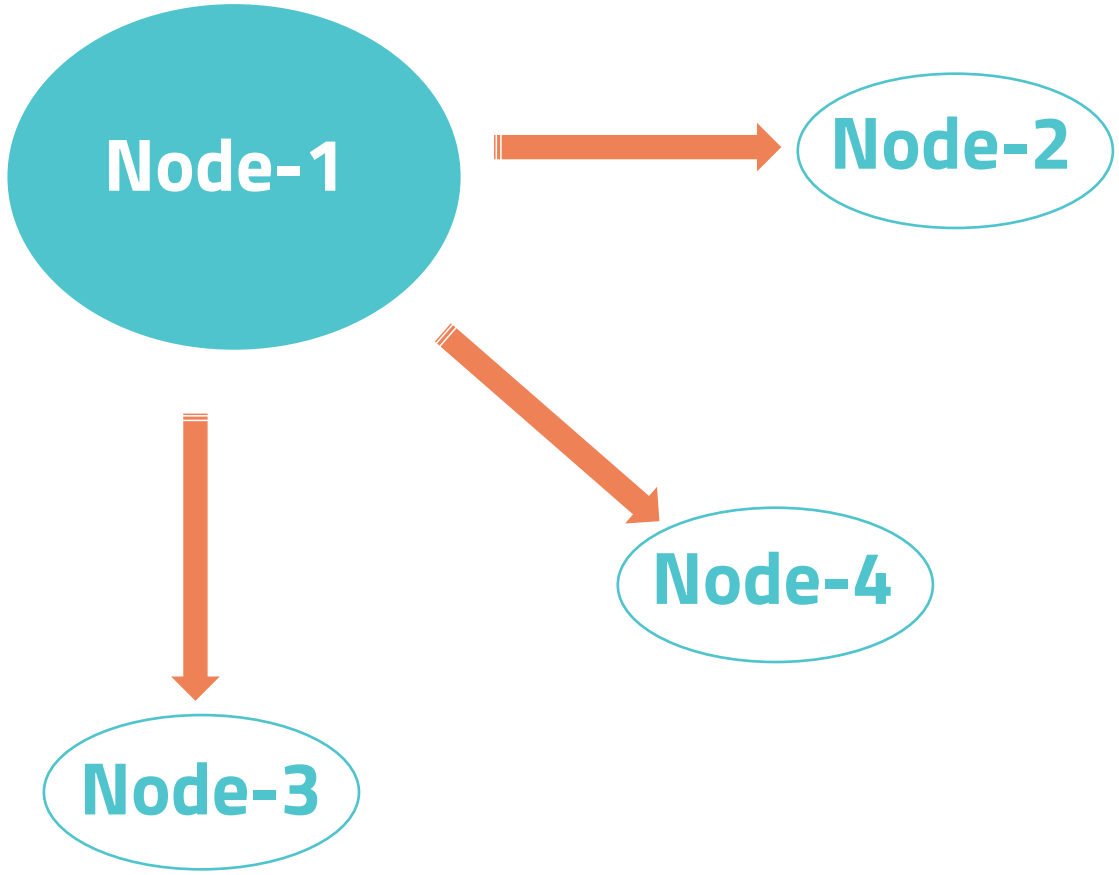
Decentralized, Distributive Ledger, Key Management System (KMS)

Common Algorithms Execute on the Same Data



AKM uses the same algorithms on the same data to re-create the same set of next session credentials on every node within the same relationship

AKM Security Relationship (ASR) Communication & Synchronization

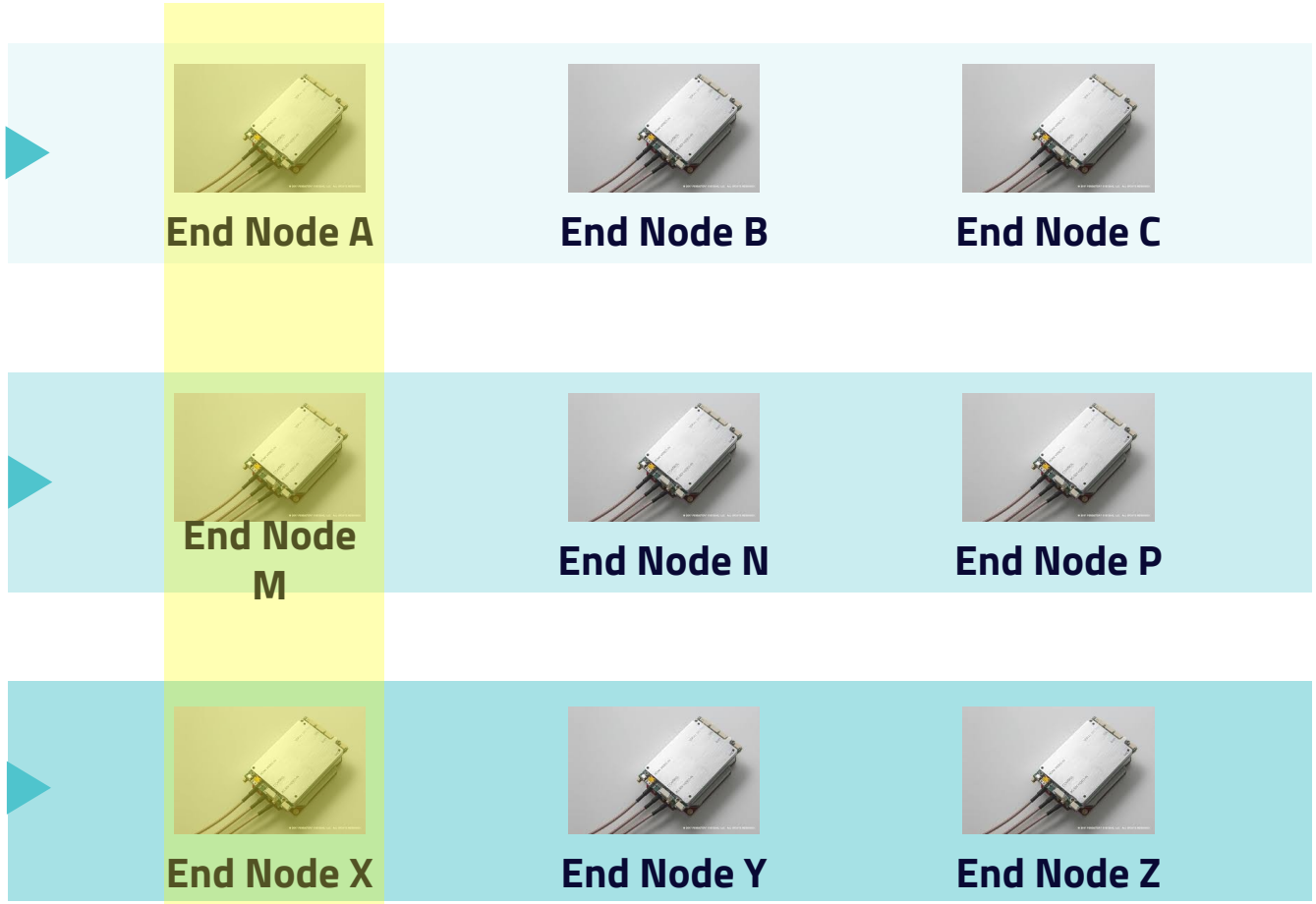


When nodes communicate with each other, they use the same set of credentials, which is how the AKM KMS supports true, multipoint end-to-end encryption.

Security Credentials are refreshed AUTOMATICALLY and AUTONONOMOUSLY every session. Thus, unlike PKI, which in a closed IoT system, may never refresh its credentials or at best, do so every 9-12 months.

Multi-point End-to-End Encryption

Overlapping AKM Security Relationships (ASRs)

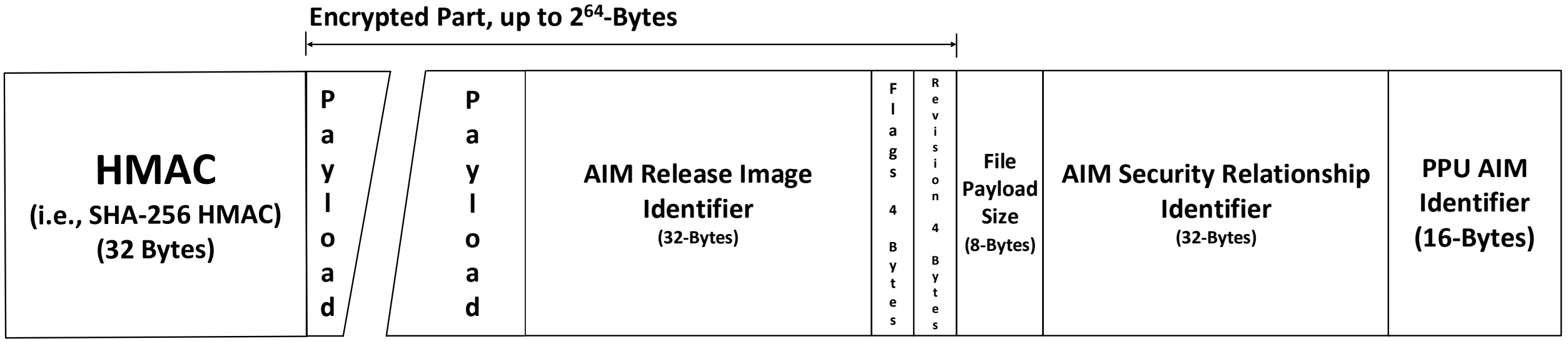


This diagram demonstrates that multiple relationships can be held by the same nodes.

In this example, Nodes A, M, and X share a security relationship (represented by the “yellow” band), but each of them have a mutually exclusive second relationship with other nodes (represented by varying shades of blue bands).

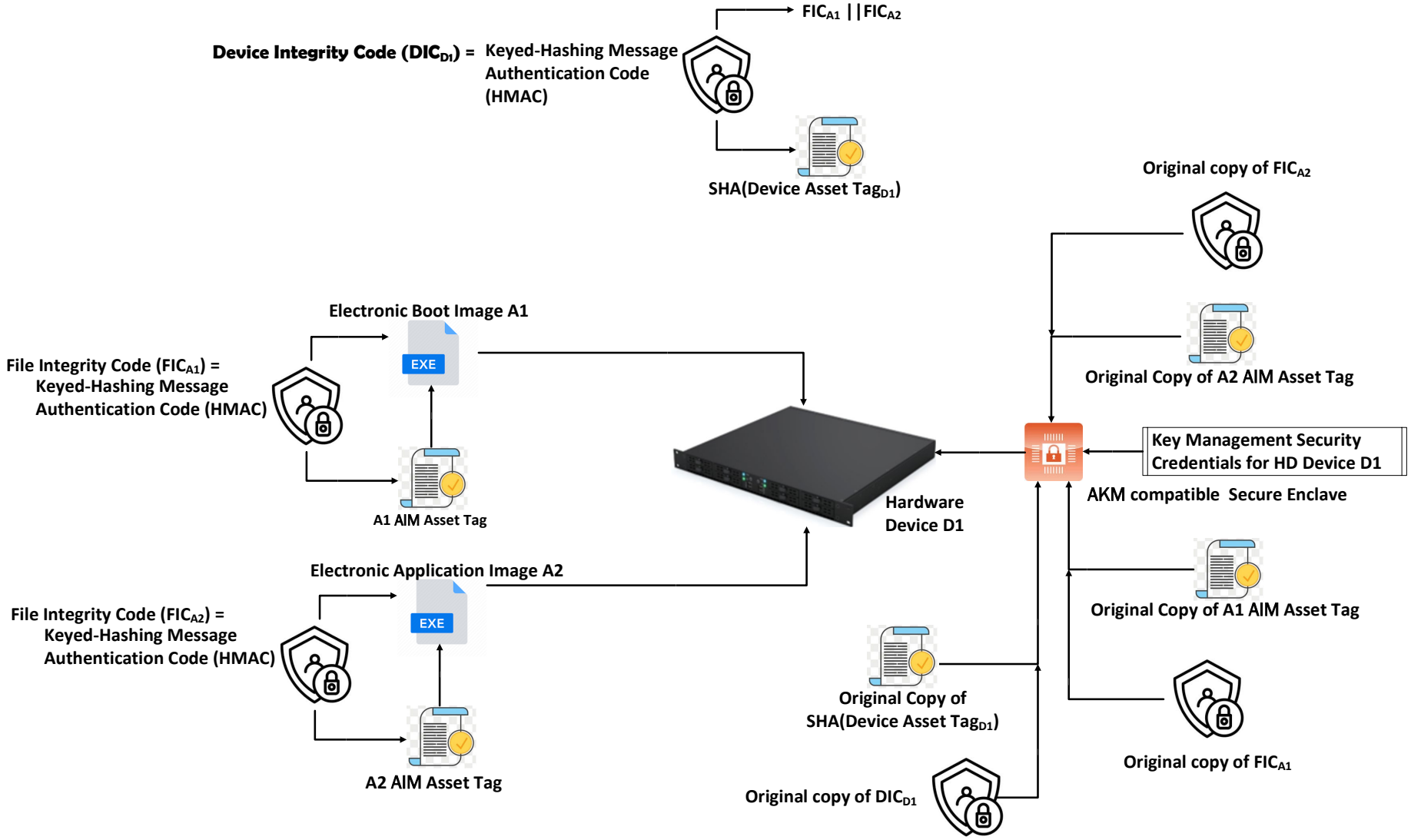
AKM KMS & Overlapping ASRs (one ledger per relationship)

Asset Integrity Management (AIM) File Structure



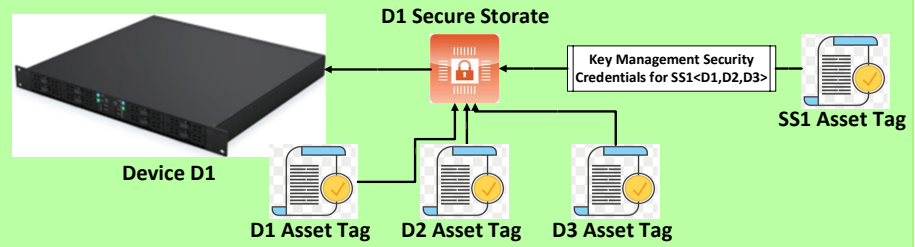
This scenario assumes an AIM security relationship between the FBCS and the Target Edge Node and could be even done via a portable static storage device, like a USB-drive. It is important to make the distinction that the AIM Security Relationship is a different type of set of security credentials than an AKM Security Relationship. The AKM Security Relationship controls "data-in-motion", while the AIM Security Relationship controls "data-at-rest" and "data-in-use".

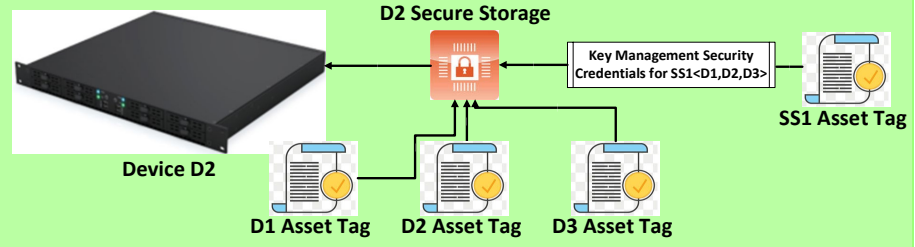
AKM/AIM Device Authentication

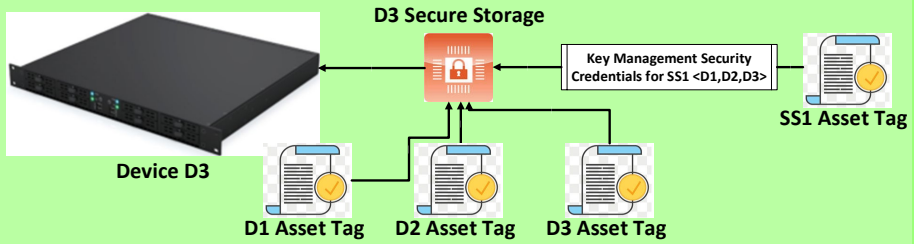


AKM Provides device authentication by binding the firmware image to the physical hardware via the secure enclave as the ZKA "Verifier"

AIM Subsystem Authentication





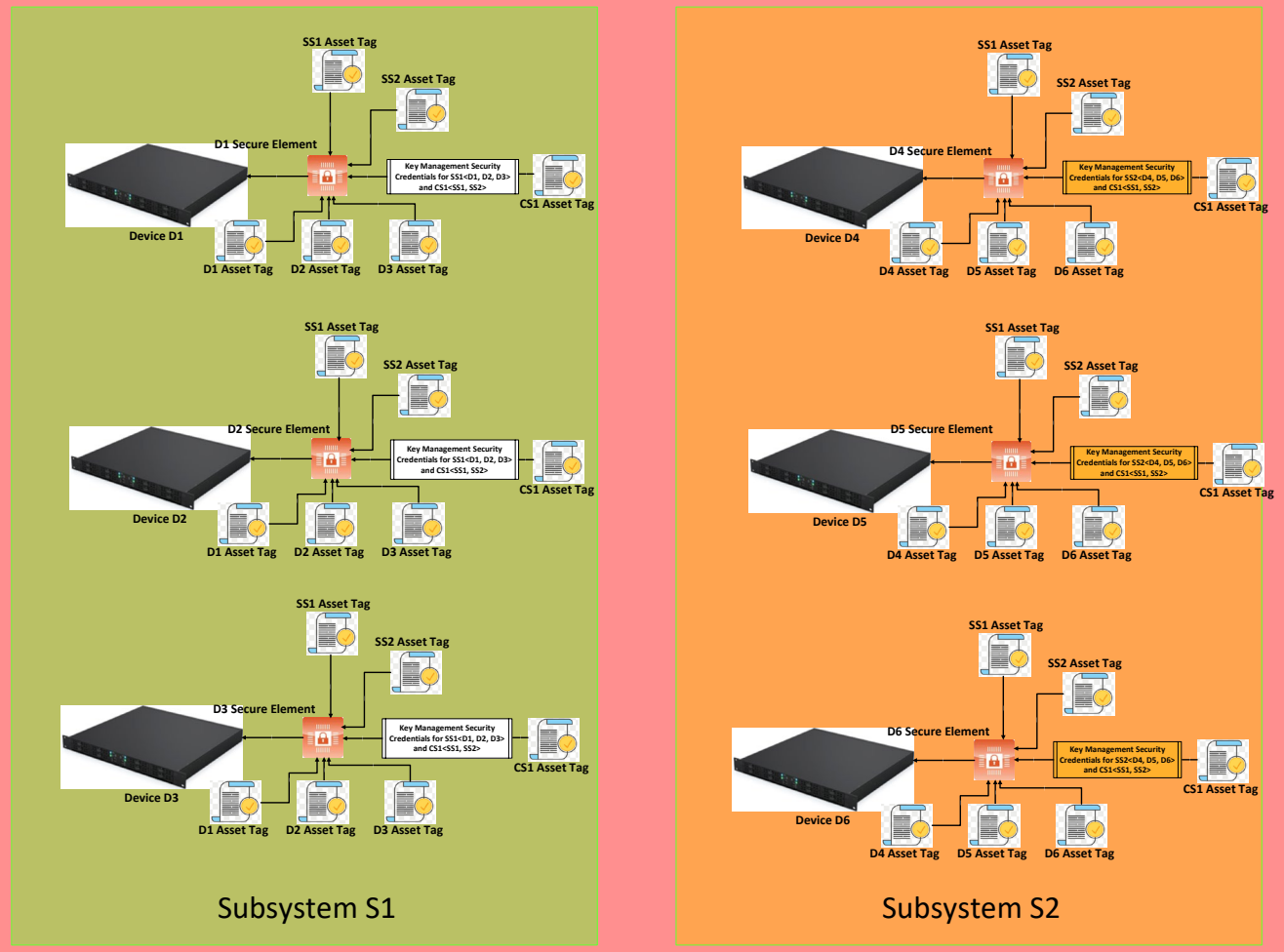


Subsystem S1

NOTE: The three devices, D1, D2, and D3 form Subsystem SS1.

Thus, each device must maintain BOTH the asset tags of the other devices (in addition to its own asset tag), plus the asset tag for its subsystem (which is calculated from aggregating the other asset tags of the individual devices within the subsystem).

AIM System Authentication



NOTE:

The two different subsystems, SS1 & SS2, form the Complete System, CS1.

Each device must maintain the asset tags of the other devices within its own subsystem, plus the asset tags of the other adjacent subsystems, plus the asset tag of the entire system.

This distributive architecture approach prevents any single point of failure, thus further ensuring the integrity of the entire system.