

Decrypted visibility in a TLS 1.3 world



**Symmetric Key Intercept for Secure
SSL / TLS Decryption in the Cloud**



Table of Contents

Executive Summary	3
The Cloud Has Won. Now What?.....	4
The New Challenges of Decrypted Visibility in the Cloud	6
Tectonic Force 1: TLS 1.3 Breaks Legacy Out-of-Band Decryption	6
Tectonic Force 2: Cloud Application Architecture Breaks Man-in-the-Middle Decryption.....	7
Tectonic Force 3: Keeping Keys & Encrypted Traffic Together Breaks Scalability.....	9
Upheaval Creates New Foundations for Decrypted Visibility in the Cloud	10
Bringing JA3 & Decryption Together.....	12
Nubeva TLS Decrypt & Symmetric Key Intercept	14
The New Solution Space: Symmetric Key Intercept Restores Out-of-Band Decryption for the Cloud	14
Symmetric Key Intercept Architecture: How It Works.....	15
The Nubeva Sensor	15
The Nubeva Controller	16
The Nubeva Decryption Engine.....	16
Automatic Updating and Secure Retrieval.....	17
Decrypted Visibility in the Cloud Restored with Symmetric Key Intercept and Nubeva	17
Packet Mirroring, Tapping and Transport.....	18
Seismic Shifts Reveal the New Foundation for Out-of-Band, Decrypted Visibility in the Cloud	19



Executive Summary

In this technical white paper, Nubeva addresses the seismic shift in decrypted cloud visibility that has created a new and massive burden caused by the clash of three tectonic forces:

1. The new TLS 1.3 encryption standard that breaks traditional out-of-band decryption due to its enforcement of perfect forward secrecy, ephemeral keys and discontinuation of certificates for decryption.
2. Cloud application architecture that is highly distributed, dynamic and decentralized which renders man-in-the-middle, firewall and proxy decryption untenable and unaffordable.
3. Decoupling keys from decryption without which decryption is effectively rendered unscalable and single-threaded.

This paper then explores the solution to these challenges and the restoration of out-of-band decryption in the cloud with the new *symmetric key intercept* architecture.

Symmetric key intercept is able to discover final, ephemeral symmetric encryption keys in any cloud workload. It is able to decrypt any cipher and any protocol. This approach uses an innovative key discovery sensor, brought to life by Nubeva, which identifies the TLS *ClientHello* in the TLS Handshake then scans workload memory using AI-based signatures to discover the symmetric encryption keys. The keys are sent through a key escrow in the client's environment and controlled by the clients IAM rules. In the escrow, keys may be preserved or purged according to the client's requirements. The escrow decouples key discovery from the act of decryption and enables massively parallel decryption operations without worrying about managing the transport of clear-text, decrypted streams from tools to tools or zones to zones. Finally, decryption happens on the tool workload and delivers both the originally-encrypted packet streams along with the newly decrypted packet streams to the tools. This allows the tools to inspect original encryption headers as well as decrypted payload. The Nubeva sensor runs as a Kubernetes DaemonSet. The Nubeva decryption engine buffers incoming encrypted streams, retrieves the appropriate symmetric key from the escrow and decrypts the packet traffic. It has the ability to read in saved files (like encrypted pcaps), forward both the decrypted and original encrypted traffic to tools, or write decrypted output to a file (e.g. decrypted pcap). The Nubeva decryption engine sits on the shared host next to each tool workload that requires decrypted packet streams for inspection and analysis which ensures that cleartext traffic is not exposed.

This architecture is brought to life in [Nubeva's TLS Decrypt](#) product.



The Cloud Has Won. Now What?

The cloud has won the day. Amazon Web Services (AWS), Microsoft Azure and Google Cloud platforms, in all their public, private and hybrid permutations, have created an environment where infrastructure provisioning and networking is elastic, on-demand and extremely fast. Ubiquitous compute availability has evolved application design patterns away from the monolithic and self-contained to the distributed and decentralized. The cloud handles the networking, resource allocation and scaling. Developers and designers are as free to pull information from on-demand third party systems through dynamic API calls as they are to spin up extra compute horsepower on-demand or create seamless redundancy in regions around the world.

The following three shifts tipped the scales to make cloud a winner:

1. IT disciplines that accompany application design and operation became cloud centric. Application architects assume infrastructure transience, horizontal scaling and topology partitioning (microservices).
2. Operations groups recognize that DevOps is table stakes, now that infrastructure availability is measured in seconds or minutes, not weeks or months.
3. IT groups use services rather than implementing components. They want the function, not the support.

This tipping point is not the moment when the public cloud became dominant, but rather the moment when that dominance became inevitable.

In the past, IT leaders had doubts about moving apps and resources off premises. But the ease of working with cloud platforms became too compelling to pass up. Instead of requiring a deep understanding about many platforms and technologies to make the data center effective, IT organizations easily interact with Azure, AWS and GCP through an API call. If using AWS Outposts, teams don't have to understand VMware or stay on top of the latest Dell initiative.

This is the new world of the cloud generation.

Highly specialized and reusable application modules are connected with on-demand networking to create the incredibly flexible and powerful systems that make businesses run.

This tipping point is not the moment when the public cloud became dominant, but rather the moment when that dominance became inevitable.



As more of the world’s application infrastructure moves to the cloud – whether public, private or hybrid – Security, DevOps and IT teams are coming to grips with the reality that they do not control the infrastructure upon which their businesses depend. Hardware and networking infrastructure belong to the cloud providers. Intra cloud and intra workload and third party API networking standards are defined by the cloud and API providers. The business consumer and application developer can accept them or not use them. The opportunity-cost of the speed and agility of the cloud is a loss of visibility and control.

As additional resources migrate to the cloud – applications, web servers, mission critical business software, labs and testing environments – security, performance and uptime become more critical. IT requires access and insight into potential threats, monitoring systems and diagnosing problems.

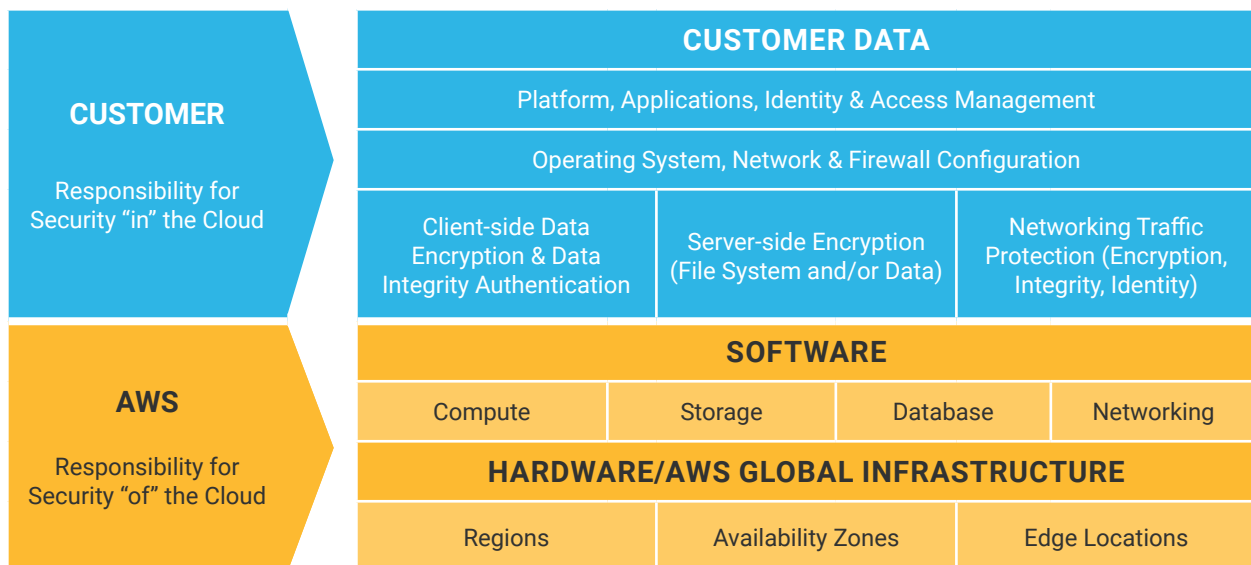
The opportunity-cost of the speed and agility of the cloud is a loss of visibility and control.

The Shared Responsibility Model

In a public cloud platform, like AWS, security and compliance is a shared responsibility between the platform provider and the user. This shared model is designed to help relieve a user’s operational burden as the platform itself operates, manages and controls components from the host operating system and virtualization layer down to the physical security of the facilities where the service operates.

AWS, for example, is responsible for protecting the infrastructure that runs all of the services offered in the AWS cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS cloud services.

Every end users’ responsibilities are determined by the cloud services they select. Typically, users manage the guest operating system, associated application software and configuration of the security group firewall. This shared responsibility offers flexibility and user control. The chart below, from AWS, shows the differentiation of responsibility commonly referred to as Security “of” the Cloud versus Security “in” the Cloud.





The New Challenges of Decrypted visibility in the Cloud

There is a seismic shift happening in the cloud. Three great, tectonic forces of change are colliding and creating unprecedented disruption for Security, DevOps and cloud professionals. Ultimately, this shift has prompted the evolution of how to gain out-of-band, decrypted visibility in the cloud. These forces are:

- Tectonic Force 1: The new TLS 1.3 encryption standard
- Tectonic Force 2: Cloud application architecture
- Tectonic Force 3: Decoupling keys from decryption

Tectonic Force 1: TLS 1.3 Breaks Legacy Out-of-Band Decryption

Against the backdrop of new application design patterns and the networking ecosystems that connect all the cloud workloads together, a new transport layer security standard has emerged. TLS 1.3 became the official encryption-in-motion standard in March of 2018. TLS 1.3 and its precursor, TLS 1.2 with Perfect Forward Secrecy (PFS), Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) ciphers and pinned certificates were designed to enforce the idea that encryption should be more robust, keys should be prolific and temporary, and decryption should only be possible by the owner of the traffic.

The new encryption standards and the older implementations of TLS that use PFS enforce three key ideas that increase security but, when combined with the cloud, create new challenges for cloud security, compliance and troubleshooting.

One-way symmetric key computation. Symmetric keys are not derived from the combination of the certificate, private key and packets. The final, symmetric encryption keys are created in such a way that there is never enough information transmitted over the wire for a snoop to derive the key. In the classic Bob-Alice-Eve scenario, Diffie-Hellman based encryption ensures that Eve will not be able to figure out Bob and Alice's shared symmetric key. This is true whether Eve is there as an attacker or as a legitimate visibility device.



Ephemeral keys. In TLS 1.3 and TLS 1.2 with PFS, final, symmetric encryption keys are for one and only one session. Each session has its own, unique key that only works on the contents of that session. After a session ends and a new one begins, the same endpoints complete a new TLS handshake and a new symmetric key is created. This means there is a massive increase in the number of symmetric encryption keys created. Symmetric keys can only encrypt and decrypt the packet contents of that one session. There is no longer any “master skeleton key” or a single “key to the kingdom.” If a key is obtained by a bad actor, it can only be used to decrypt the one set of packets from the session for which it was created.

Decryption Requires Participation in the TLS Handshake. Because of the way that PFS works and its enforcement in TLS 1.3, only those endpoints that participate in the TLS handshake have the information required to decrypt the packets sent between those endpoints. In TLS 1.3, certificates are used only for authentication. Everything after the *ServerHello* is encrypted. Therefore, the contents of the certificates are encrypted in the TLS handshake, and consequently, not available for use as components of key construction or derivation. This also means that a decryption solution must only be present on one or the other end of a TLS handshake in order to have access to the data flowing between those two systems (at least for that session).

The impact is that *TLS 1.3 breaks legacy out-of-band decryption*. Because certificates are not available for decryption key derivation, old solutions do not work. In the past, all the session traffic between two points could be decrypted once the encryption key was provided or derived. Now keys are ephemeral; they work only for a single session. Legacy, out-of-band solutions that relied on RSA key exchange or certificate access for decryption do not work in the new TLS 1.3 world.

We have seen how the new TLS 1.3 standard breaks legacy out-of-band decryption that relies on RSA key exchange and certificate inspection. But what about in-line proxies and man-in-the-middle decryption?

The impact is that TLS 1.3 breaks legacy out-of-band decryption. Because certificates are not available for decryption key derivation, old solutions do not work.

Tectonic Force 2: Cloud Application Architecture Breaks Man-in-the-Middle Decryption

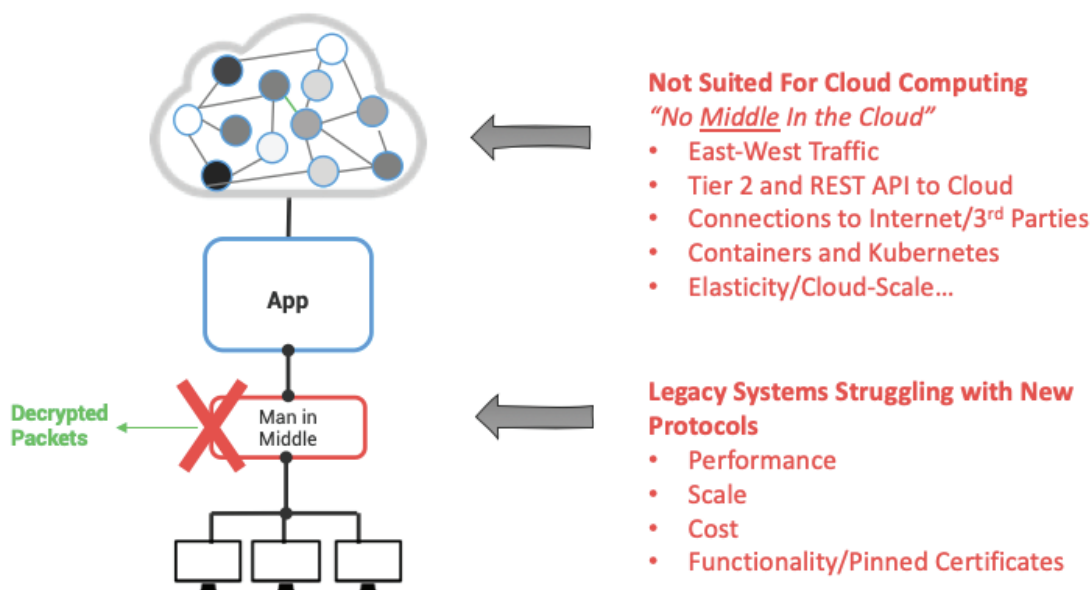
Applications are no longer single, monolithic code structures. The cloud has opened up the ability to distribute and decentralize application layers and processes. This distributed and decentralized, microservice-based architecture means that applications are an amalgam of networked services, API calls and elastic workloads. Modern application workloads communicate over TLS encrypted networks to perform their tasks, deliver data and run mission critical applications for business. Each workload makes thousands of TLS-secure connections each day. Each workload is a TLS client and sometimes a TLS server.



Historically, the only option to gain decrypted visibility was to decrypt at the server or create chokepoints with man-in-the-middle architectures and decryption zones. This was acceptable in the data center where East-West connections were controlled, TLS clients and TLS servers were known and network edges were hardened perimeters. Incoming and outgoing communication – North-South – was an obvious location for inspection, monitoring and control. In the cloud, the assumptions of known perimeters, full control of East-West connections and complete control of North-South ingress/egress points do not hold.

The impact is that, as containers and microservice-based architectures accelerate the decentralization of application workloads in the cloud, *there is no longer a “middle” into which a decryption solution may be inserted.* The cloud does not tolerate in-line solutions precisely for this reason.

Man-in-the-middle decryption offered by some legacy firewalls and inline security devices either don't work in the cloud or require restrictive architectural designs. Imagine trying to jam a decrypt-capable firewall in between each connection in a scale set. Imagine trying to pay for a firewall doing MITM inspection and proxying in between every back-end third party API connection for an application. Imagine the architectural nightmare when trying to run a 10Gbps duplex connection through a 5Gb firewall chokepoint. You would need three firewalls at the choke point to cope with peak load and that's before any scaling events. It is clear that the cloud simply will not tolerate in-line, man-in-the-middle solutions for decryption and visibility.



To be clear, any network or security architecture that sacrifices the inherent elasticity and distribution of cloud workloads – like hair-pinning all connections through a firewall or packet broker bottleneck – is a bad and broken architecture. When a tool or technique requires that the cloud's elasticity is eroded then the value of the cloud is lost, expenses are increased and flexibility is diminished.

We have seen how the new TLS 1.3 standard breaks legacy out-of-band decryption that relies on RSA key exchange and certificate inspection. We have seen how modern cloud architecture does not tolerate in-line, MITM decryption architectures. A third force is a new force that is born in the cloud and emerges uniquely from the requirements of TLS 1.3 and the capabilities of cloud architectures.



Tectonic Force 3: Keeping Keys & Encrypted Traffic Together Breaks Scalability

In legacy tools decryption happens when a device receives encrypted traffic, calculates or receives the static (not ephemeral) key and then decrypts the traffic which it can then inspect or forward on as clear text to other tools. In each case, the keys and the encrypted traffic are bound together in the same processes. This effectively makes the decryption process single-threaded. The reason keys and encrypted traffic are kept together as long as possible is to preserve the security of the encrypted traffic and to lower the risk of splitting up the keys – which in the old world of TLS 1.2 and before, can be used to decrypt anything from any point in time sent between the two endpoints that created that key.

While the actual process of decrypting traffic is quick, the process of key derivation is time consuming and resource intensive. A Gartner study reports that enabling decryption on leading next-generation firewalls can degrade firewall performance by as much as 80% and reduce the transactions per second by more than 90%.¹ It is not uncommon for these statistics to be hidden because they are not flattering.

John Maddison, writing for Network Computing.com puts it this way: “According to recent test results from NSS Labs, very few security devices can inspect encrypted data without severely impacting network performance. On average, the performance hit for deep packet inspection is 60 percent, connection rates dropped by an average of 92 percent and response time increased by a whopping 672 percent. Even more concerning, not all products were able to support the top 30 cipher suites either, meaning that some traffic that appeared to be analyzed wasn’t being processed by some of the security devices at all.”

On average, the performance hit for deep packet inspection is 60 percent...

The truth is that TLS handshakes are computationally complex and can eat up system resources. TLS 1.3 reduces some of the computation load by decreasing the number of round-trips in the handshake. But the challenge of computation for ephemeral, session-by-session symmetric keys is still huge on man-in-the-middle decryption architectures.

Compounding the challenge for the legacy decryption approach is the fact that most IT and security teams struggle with the following challenges:

- They have multiple tools that need to see decrypted traffic, which causes a significant decrypt – re-encrypt – forward burden on the decryption tools and the network overall.
- The MITM handshake is really a double handshake which exacts a large CPU tax (as described above) making it inefficient to run this function on multiple security tools. MITM terminates the inbound handshake, decrypts, inspects and then recreates the handshake on the other side.
- The serial chaining of multiple security tools together while scaling the decryption process is difficult.
- Properly handling, protecting and maintaining the isolation of clear-text traffic for regulatory compliance is difficult.²




The impact is that, like the problem with man-in-the-middle decryption, keeping the keys and encrypted traffic together breaks scalability of visibility for security, DevOps, compliance and cloud systems. The old approach creates many problems for performance, data handling and cost, while solving only one – keeping the original traffic encrypted for as long as possible.

We have talked about forces that are shifting the foundations of decrypted visibility in the cloud. We have seen how the new TLS 1.3 standard breaks legacy out-of-band decryption that relies on RSA key exchange and certificate inspection. We have seen how modern cloud architecture does not tolerate in-line, MITM decryption architectures. We have seen how keeping keys and encrypted streams single-threaded is required by the new TLS standards and how that breaks the scalability of the cloud.

Upheaval Creates New Foundations for Decrypted Visibility in the Cloud

These three tectonic forces have collided in the cloud to create upheaval. This upheaval creates new groundwork, a new environment for a new approach to decrypted visibility in the cloud. The new approach must address each of the three forces described above.

1. Any new approach to decrypted visibility in the cloud must support TLS 1.3 with its certificate encryption, enforced strong ciphers, ephemeral keys and enforced PFS. It must also be backwards-compatible with earlier TLS versions that enable perfect forward secrecy.
2. A new approach must be out-of-band so that modern cloud application architectures can scale with all the potential the cloud offers and make use of microservice design patterns, containerization and third party API data that come with their own controlled encryption and pinned certificates.
3. A new approach to decryption should decouple key identification and decryption. By decoupling key discovery.



Additionally, there are several factors of the new cloud environment that any new decryption approach should include.

- 1. Cloud Native.** Any new approach should be built from the ground-up as a cloud native solution. Legacy solutions masquerading as cloud-ready have merely been lifted and shifted from the data center. These solutions may run in the cloud but lack the fundamental design paradigms that deliver cloud benefits. Running monolithic applications and execution contexts in a VM or bloated container is not the same as a small footprint, extremely nimble, cloud-native design. A true born-in-the-cloud solution will take advantage of microservice architectures, containerization, scalability and easy spin-up / spin-down elasticity. A solution that scales at cloud levels will work as well for the single user troubleshooting as for an enterprise-wide monitoring deployment. A solution that is cloud-native will not require encryption library locations to be known and set ahead of time as this only limits scalability and elasticity in the cloud.
- 2. Open.** Any new approach should be able to work with any packet mirror or tap source, any packet brokering source, and any tool destination source. A truly open solution will not require users to know in advance where the encryption and TLS libraries are stored in each application and will not require that only certain ciphers and certificates are used. The days of tool and vendor lock-in are over. Best-of-breed means the openness and flexibility to select the tools, processes and platforms that are best for your business rather than bending your business around inflexible, vendor-established requirements.
- 3. Universal.** Any new approach must be able to handle any cipher, any TLS standard, and any protocol. Artificial limitations on visibility are half measures that can leave you fully vulnerable. True universality can be tested and proven without requiring turn-down of TLS, without omitting certain ciphers and without requiring application modification.

Bringing JA3 & Decryption Together

Many companies are foregoing packet-level decryption in the cloud thinking that is simply isn't possible in the new TLS environment. Some are even being told by vendors that decryption is not necessary and that they can use JA3 fingerprints to inspect their TLS 1.3 (and other TLS) traffic. As we have seen, it is not only possible but also ideal to inspect fully decrypted packet traffic. While JA3 and JA3S fingerprinting can be useful as a first-pass inspection process, it can only be counted on to find the obviously errant and anomalous connections. JA3 must be paired with full packet decryption in order to provide complete TLS security.

JA3 is not a solution that lets you see the unencrypted packet content. It is another version of metadata analysis. It also carries some substantial limitations that further restrict the use cases where it is viable without being paired with full decryption.

Studies consistently show that companies have very low packet level visibility in the cloud.

- Only 15% of companies have the data they need to properly monitor public cloud environments.
- 95% say visibility problems in the public cloud lead to application or network performance issues.
- Less than 20% say they have complete, timely access to public cloud packet data.
- 93% say packet level visibility is critical for security.

In order to understand how to best deploy JA3, it is important to understand exactly what JA3 is. JA3 uses the unencrypted attributes of the TLS *client hello* to create a fingerprint that can possibly identify potentially harmful or unusual clients and traffic. JA3 does not have access to the encrypted payload which includes the certificate and the packets in TLS 1.3. JA3 does not decrypt the traffic.

JA3 looks at 5 parameters of the *client hello* including the TLS version, allowed ciphers, extensions, allowed elliptic curves and allowed elliptic curve formats. JA3 then combines these values and creates an MD5 hash of them which becomes the JA3 / JA3S "fingerprint". This fingerprint is sometimes enough to identify various applications that are the source of the traffic. This fingerprint can be compared to known signatures of bad, anomalous or unusual systems, environments or connections. JA3 based detection makes use of the fact that servers tend to (but do not always) respond to requests from the same client in the same way. This predictability allows for fingerprint hashes to be compared to signatures that may represent indicators of compromise or other anomalies.

There are challenges with JA3 when it is not also paired with full decryption. First, the SSL/TLS libraries that are used in a connection change the field values in the TLS *client hello* that JA3 uses to create its signatures. OpenSSL will have different values than will BoringSSL, NSS, Apple Secure Transport and Microsoft SChannel. Different *versions* of

the same library as well as different libraries all create different combinations of fields and values that result in different JA3 fingerprints. The result is that the same malware connecting to a rotating command-and-control server using different TLS libraries will result in different JA3 fingerprints. This creates a moving target for fingerprint identification. However, when paired with full decryption, a positive “hit” on one of the resulting fingerprints could trigger a process where full decryption is turned on for the suspicious connections. Instead of waiting to compile enough potential positives with JA3 alone to trigger an escalation and investigation, decrypting the suspicious traffic out-of-band will deliver rapid validation of the potential threat. In this way JA3 can act as an early warning system for a robust, out-of-band, decryption-based threat investigation process.

Second, the JA3 fingerprint is delicate. The goal is to take a known bad fingerprint - for example, a fingerprint that indicates a TOR client is being used or that malware is doing the network communication – and look for a match to that fingerprint amongst the fingerprints created on your network. This is a signature-based approach. However, because the fingerprints are dependent on the TLS *client hello*, they can be quickly outdated by different OS versions, different TLS versions, whether or not the malware was compiled on the victim machine, compiled with an office macro or any number of other factors that will change the fingerprint. This is why JA3 alone is insufficient to deliver visibility into encrypted network traffic. However, when included as the front-line passive detection layer in an inspection process that includes full network traffic decryption, it can provide good first-line of defense. Much like a firewall can block known bad websites, JA3 can be used to detect known bad fingerprints and pass the rest of the traffic through to the passive decryption layer that can send the rest of the decrypted traffic to an IDS, DLP or other DPI solution.

Finally, to work as indicators for security and troubleshooting, JA3 fingerprints must be unique to each client. Tests have shown that as much as 42% of JA3 fingerprints were different for the *same* clients. This would indicate a false positive, there should be 0% difference. Some tests have shown that as much as 76% of JA3 fingerprints for *different* clients were actually the same. This would indicate a missed alert or a hole in security since each client fingerprint should be unique. To be useful for security, JA3 needs to be backed up with full, out-of-band packet decryption and inspection.

Similar to NetFlow and log-based analysis, JA3 is an important arrow in the security professional’s quiver; but it is not the only arrow and should never be relied upon as the sole solution. Metadata based analysis may be able to tell you that there is a party going on in a certain house on a certain street if you know enough to look in that neighborhood. It cannot tell you who is at the party or what they’re doing inside the house. Is it a dinner party or a flash-mob? Were the people invited or are they crashing? To see who is at the party, you need access to packets. In many organizations that rely only on JA3 or other metadata for security monitoring, they may see that a party is going on and then they turn on packet capture in hopes of being able to get a peek inside the house while the party is in full swing. Unfortunately, by the time pcaps and tcpdumps are running, the party is over. This means the organizations have to wait and hope that a party in the same party in the same house happens again so they can catch a glimpse of who is there.

The ideal solution is pair existing log and metadata analysis with symmetric key discovery and on-demand, scalable, stream decryption so that, when JA3 says a party might be going on, you already have the literal inside information.



Nubeva TLS Decrypt & Symmetric Key Intercept

The New Solution Space: Symmetric Key Intercept Restores Out-of-Band Decryption for the Cloud

Nubeva has created a new solution for out-of-band decryption in the cloud. The process is called *Symmetric Key Intercept*.

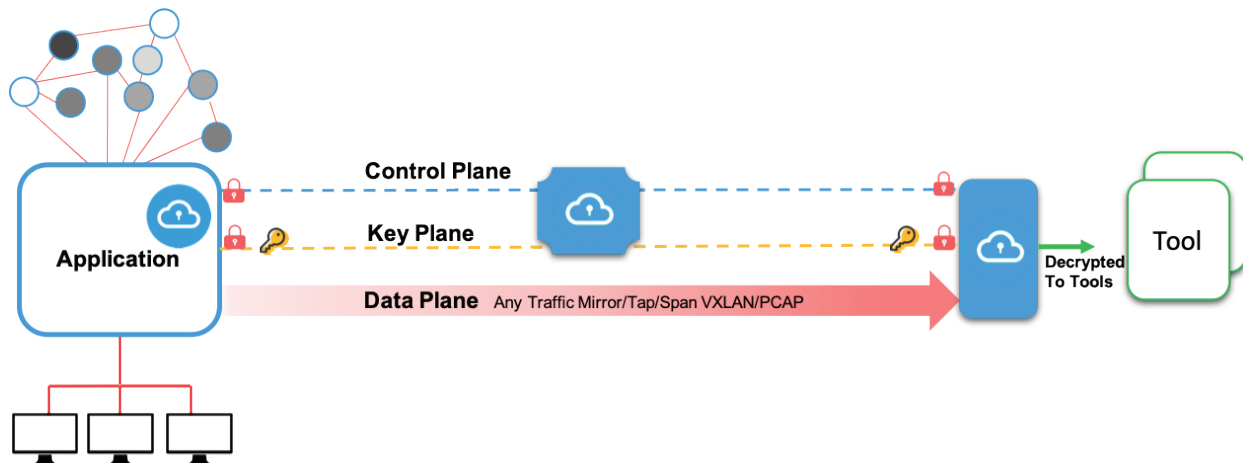
Symmetric Key Intercept is a patent-pending process that works by separating and solving the three shifts of cloud decryption:

1. Discovering and obtaining the final, ephemeral, symmetric encryption key.
2. Out of band decryption of particular, encrypted, packets from a replicated stream or stored file (pcap).
3. Decoupled key discovery and decryption.

This cloud-native architecture delivers universal TLS visibility and decryption for any workload whether it is acting as the TLS server or TLS client. Symmetric Key Intercept works after the TLS Handshake by retrieving the final, ephemeral, symmetric encryption keys from workload memory. This means that it works for any cipher, on any protocol. It works with perfect forward secrecy and it works with any TLS / SSL standard – including the new TLS 1.3. This architecture enables real-time, multi-destination, decentralized decryption of mirrored traffic as well as instant decryption and replay of mirrored and encrypted pcaps that can be stored for future investigation, compliance or inspection.

Symmetric Key Intercept architecture answers the secure vs. visibility conundrum that most enterprise IT organizations need to solve. The process ensures that original end-to-end encryption is preserved while cloud-scale decrypted visibility is created.

Symmetric Key Intercept Architecture: How It Works



The Nubeva Sensor

First, AI rules-based, final key discovery, and extraction happens at either end of the TLS handshake. Nubeva calls these rules, *signatures*. A Nubeva sensor runs as a Kubernetes DaemonSet that is deployed in source groups in your cloud environments. A DaemonSet ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

With negligible (<0.1) CPU impacts and read-only permissions, the Nubeva sensor discovers and extracts the final ephemeral symmetric encryption keys. It does this by extracting keys from core memory. The Nubeva sensor identifies and triggers on TLS client "Hello" for TLS Handshake. It then analyzes mem- changes with AI-based heuristics, called *signatures*, to discover keys. The key discovery signatures are loaded from the Nubeva controller, which makes it updatable and not library dependent. This also means that it discovers the symmetric keys for all session types and all protocols. This makes it extremely extensible to other applications. Finally, the Nubeva sensor is able to perform key identification and extraction for most cloud workloads including containers, microservices running in containers and VMs; all without requiring configuration settings or workload restarts.

Because the symmetric key intercept approach happens after the *TLS handshake*, it works whether the sensor is on the TLS client or TLS server. This allows for symmetric key intercept even when the session, TLS level and ciphers are initiated by a third party – for example from an API call. This TLS client/server approach is critical for universal, decrypted visibility in cloud environments where applications are made of decentralized, distributed workloads and third-party data feeds. Because TLS 1.3 enforces perfect forward secrecy and ephemerality of encryption keys, there can be hundreds of thousands of symmetric keys created each day. Ephemerality means that a new, symmetric key is created for each session. Symmetric key intercept from Nubeva discovers and extracts each and every symmetric key as it is created in memory.



The Nubeva Controller

Next, once the symmetric keys are extracted, they may be sent directly to the decryptor for use or securely stored within the enterprise's own cloud in a secure key depot. The symmetric key depot keeps all your keys available for as little or as long as needed. Security and troubleshooting tools like Wireshark, Moloch, Zeek or commercial tools can perform decentralized and scalable decryption when and where needed; in real-time, in parallel and even on historical replays of saved pcaps. The key depot is the customer's asset and never touched, managed or owned by Nubeva. Access to the key depot is fully controlled by the client's IAM policies. Nubeva provides a cloud formation template to create a key depot with AWS DynamoDB, a highly secure Amazon database. However, customers may use their own solution to house the key depot. The Amazon DynamoDB, provided for POC and testing, delivers encryption at rest out of the box. This provides enhanced security by encrypting all of your data at rest using encryption keys stored in AWS Key Management Services (AWS KMS). The symmetric keys never leave the enterprise's environment. The enterprise retains complete control and keys are never exposed to hackers or bad actors. Keys may be purged or preserved for as little or as long as needed.

The key depot is a fundamental concept for Symmetric Key Intercept. Architecturally, the presence of a secure symmetric key depot enables highly scalable, low-cost strategies. Because the depot is separate from the sensor or decryptor, it enables both decrypt now and decrypt later strategies. It allows for any decryptors, not just Nubeva's decryptor. It enables massive, parallel decryption as well.

When coupled with encrypted stream replication, it allows for simultaneous inspection, monitoring and troubleshooting at each of the separate tool destinations. It is this capability that allows for the decoupling of key discovery and decryption. The decoupling of key discovery and decryption allows for massively parallel scaling for decryption, investigation, root-cause analysis, IDS, DLP and even IT troubleshooting as we'll see below.

The Nubeva Decryption Engine

Finally, a software decryption container sits next to each tool destination workload. The decryptor buffers incoming encrypted packet traffic, retrieves the correct key from the key depot and decrypts the traffic. The decryption engine then feeds the decrypted packets to the tool destination along with the original, encrypted traffic stream. Optionally, teams may configure decryption engines to write out the originally encrypted and newly decrypted packet streams to a file for secure storage, compliance inspection or forensics.

In this way, tools, teams and processes have access to the original, unaltered encrypted traffic streams in addition to the newly decrypted streams. They are able to inspect the encrypted traffic headers as well as the decrypted packet payload. Decryptor engines handle the synchronization of keys with packet flows, assuring that all the traffic received is matched with keys, and is fully decrypted.

There is no MITM set up and no TLS intercept-decrypt-re-encrypt-and-send-on process. This preserves original end-to-end encryption without introducing the additional vulnerabilities – associated with HTTPS interception and MITM TLS proxies – from downgrading encryption or poor certificate verification. This architecture also reduces cost and performance impacts typically seen in in-line MITM set ups.



Automatic Updating and Secure Retrieval

When any instance containing a Nubeva sensor or a decryptor engine launches, the sensor can automatically connect to the Nubeva TLS Cloud Console and register itself. Registering allows it to automatically obtain configuration, signature and software updates when they are available. Sensors use HTTPS to make REST API calls to the cloud console. Control traffic always originates at the sensor. Data plane traffic (mirrored filtered traffic) is routed based on the users' network configurations. Mirrored packets are never sent to the cloud console. The control plane does not directly modify, nor does it require the user to modify networks or security settings, save for allowing outbound HTTPS (TCP port 443) from subnets containing cloud sensors.

Decrypted Visibility in the Cloud Restored with Symmetric Key Intercept and Nubeva

The new Symmetric Key Intercept architecture created by Nubeva ensures decrypted traffic is never exposed to potential threats if it gets intercepted. Instead of decrypting traffic in storage then sending it to monitoring tools for inspection, Symmetric Key Intercept allows users to send encrypted traffic to tools, databases or storage and then decrypt right at the tool. The architecture is easy to deploy, and scales to meet any traffic load without any configuration overhead or architectural constraints.

With Symmetric Key Intercept in place, cloud DevOps and security teams can, with confidence, decrypt TLS traffic inside their cloud environments – enabling security, performance, and diagnostic systems and processes.

Imagine that a SOC team has three or more tools – each that need decrypted visibility to the same packet stream. One tool might be an IDS. One might be a DLP solution. One may be an APT or forensics or even a tool for IT troubleshooting. By decoupling key discovery and decryption, the encrypted packet stream can be mirrored or replicated to each of the three tools at the same time. With each of these fronted by a Nubeva decryptor sensor, each tool and analyst running that tool can perform their duty at the same time. There is never a need to manage the hand-off of a decrypted packet stream in a serial set-up as in a decryption zone. There is never the need to orchestrate the security of clear-text packet traffic since that clear-text is never passed around, decryption happens right on the tool.



Packet Mirroring, Tapping and Transport

Nubeva TLS Decrypt is independent from, and a complement to, any existing and emerging packet acquisition and brokering solution. Organizations making use of cloud provider infrastructure taps like Microsoft Azure VTAPs or Amazon VPC traffic mirroring continue to use them with no interruption and no changes. Organizations using any third party solution like Ixia, Gigamon or Bigswitch can continue to use those systems, uninterrupted for packet acquisition and brokering. DevOps and IT support groups making use of TCPReplay from mass-stored files can also use Nubeva's TLS Decryption solution without any change to the way they work – except that now they can immediately see the previously encrypted packet traffic.

Nubeva's breakthrough TLS 1.3 Decryption invented the only enterprise-ready and commercially viable, Symmetric Key Intercept solution. It is born in the cloud and purpose-built for the cloud generation. It is architected for and runs as a native cloud solution. It is distributed, modular and loose-knit. Customers can have it up and running in minutes and tailor it to their specific needs.

Because key acquisition is independent of decryption, Nubeva supports cloud workload refresh, "rehydration" and "restocking" models. Because it is a lightweight and unobtrusive observer to the TLS handshakes, stops, starts, removes, adds or even crashes of the sensor do not impact the workload against which it runs. In the event that the key sensor stops or crashes on the tool workload, packet traffic is bypassed and continues on as it would otherwise.

This modular approach delivers both performance and cloud scale. Users are able to scale from one to millions of systems. The Nubeva sensor has been successfully tested to more than 10,000 keys per minute. In parallel decryption configurations (where one encrypted stream is replicated to many systems with the sensor) customers see equivalent decrypted throughput on an order of magnitude of terabytes per second. This open system approach is based on the architecture and paradigm of the public cloud. Nubeva TLS Decryption supports any source, any packet structure, any tool destination and any usage scenario.

Nubeva TLS Decrypt

- Born in the cloud
 - Enterprise-ready
 - Distributed
 - Modular
 - Loose knit
-



Seismic Shifts Reveal the New Foundation for Out-of-Band, Decrypted Visibility in the Cloud

The new Symmetric Key Intercept architecture ensures decrypted traffic is never exposed to potential threats if it gets intercepted. Instead of decrypting traffic in storage then sending it to monitoring tools for inspection, Symmetric Key Intercept allows users to send encrypted traffic to tools, databases or storage and then decrypt right at the tool. The architecture is easy to deploy, and scales to meet any traffic load without any configuration overhead or architectural constraints.

With Symmetric Key Intercept in place, cloud DevOps and security teams can, with confidence, decrypt TLS traffic inside their cloud environments – enabling security, performance, and diagnostic systems and processes.



Nubeva's first-of-its-kind solution helps put to rest the concerns of visibility and security in the public cloud.

For more information, visit: www.nubeva.com/products

Sources and Citations

¹ "Protecting from a Growing Attack Vector," 2016, <https://www.gartner.com/imagesrv/media-products/pdf/radware/Radware-1-2Y7FROI.pdf>

² "The Role of Network Packet Broker (NPB) with SSL decryption in Network Monitoring," July 28, 2019, <https://pupuweb.com/npb-ssl-decryption-network-monitoring/>