2024-10-23

클라우드 네이티브 환경에서 바라보는
암호화 체계 가시화 기술 및 암호 전환 방안

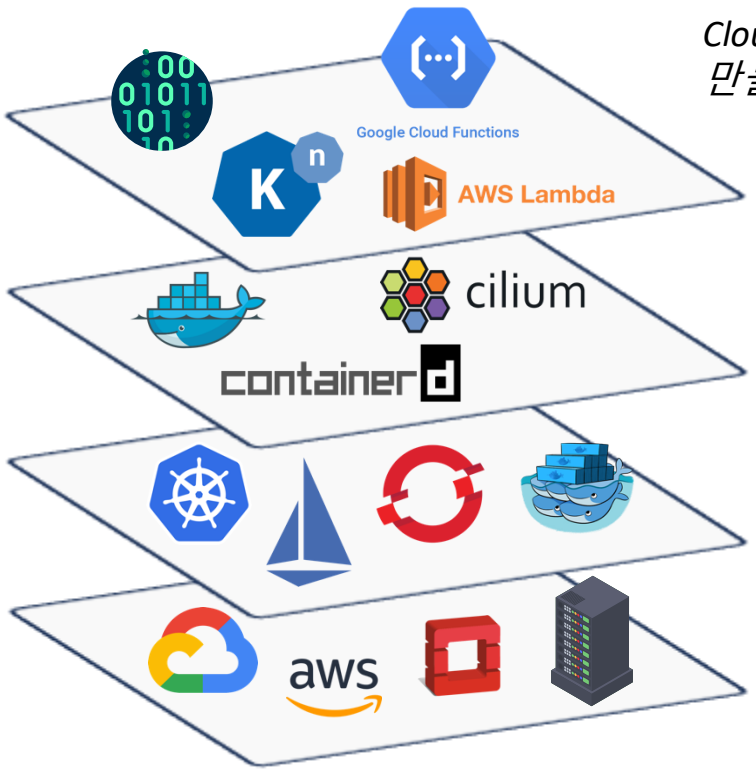
남 재 현
단국대학교 컴퓨터공학과

단국대학교 DANKOOK UNIVERSITY
BoanLab Networked Systems and Security

목 차

클라우드 네이티브 환경

클라우드 네이티브 환경



Cloud-Native

Cloud 환경의 특성을 고려하여
만들어진 애플리케이션 및 시스템

Code
애플리케이션 코드 실행

Container
컨테이너 애플리케이션 구동

Cluster
컨테이너의 컴퓨팅 자원 제공

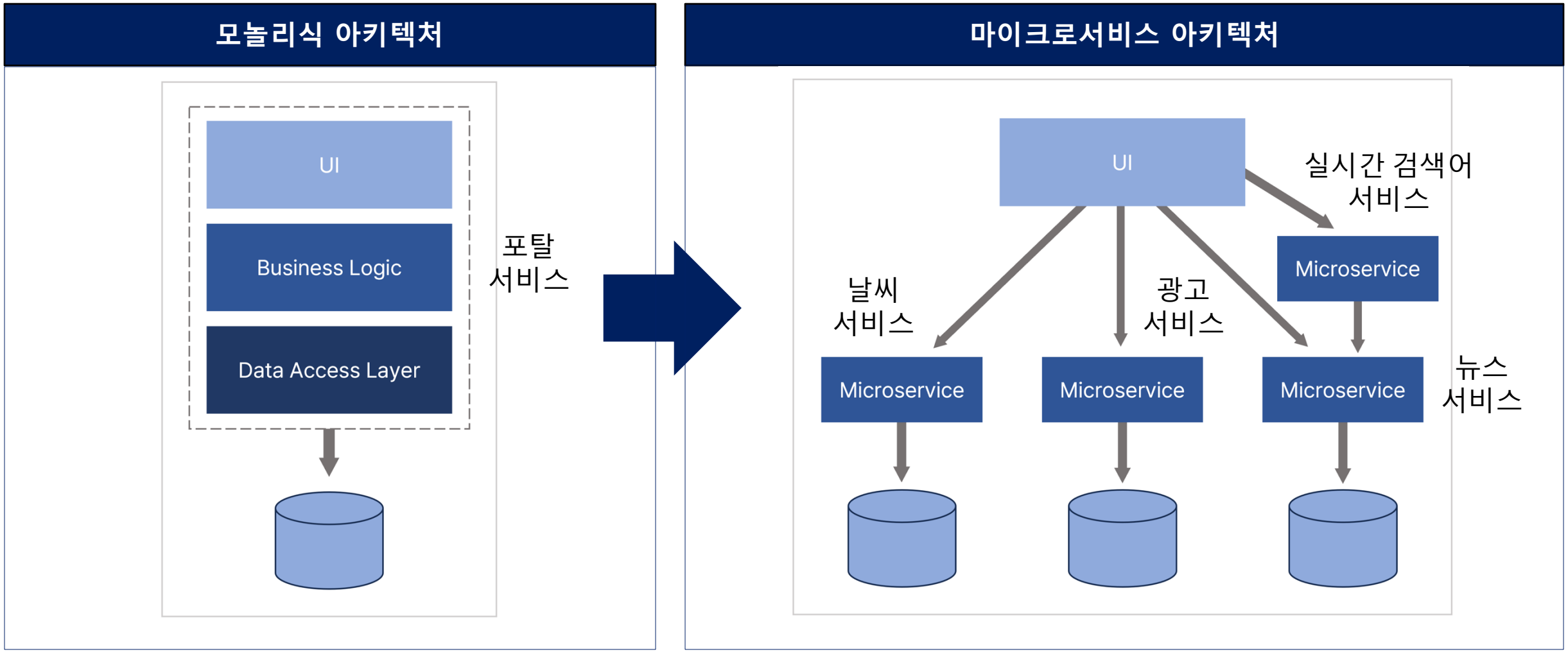
Cloud
서버/네트워크 등 인프라 자원 제공

클라우드 네이티브의 장점

마이크로서비스 구조에 따라 구현하며
경량화된 컨테이너로 실행하여
개발, 통합, 테스트, 배포가 **지속적**으로 이루어짐

민첩한 애플리케이션 개발 및 배포
빠르고 효율적인 서비스 제공 가능
높은 유연성, 확장성, 비용 효율성

클라우드 네이티브 환경



클라우드 네이티브 환경

클라우드 네이티브로 환경으로의 전환

1단계: **레거시**	2단계: **클라우드**	3단계: **클라우드 네이티브**	
물리(온프레미스)	IaaS	PaaS + 컨테이너	MSA + 컨테이너
		DevOps + CI/CD	
	Lift & Shift (OS, 데이터, App 등을 그대로 옮기는 것)	**Re-Factor** (클라우드 환경 제공)	
기존 애플리케이션	클라우드 준비단계 애플리케이션	클라우드 친화단계 애플리케이션	**클라우드 네이티브 애플리케이션**

출처: NIA 재구성

국내외 클라우드 네이티브 전환 사례

테크42
https://www.tech42.co.kr › 아마존웹서비스-100억-달...

아마존웹서비스, 100억 달러 'NSA 클라우드 계약'... MS 제쳐

2022. 5. 2. — Amazon Web Services was awarded a National Security Agency cloud computing contract worth up to $10 billion. The contract, code-named ...

미국 NSA

Amazon Web Services
https://aws.amazon.com › blogs › publicsector › tag › u...

U.S. Department of Defense | AWS Public Sector Blog

AWS announced **AWS** Modular Data Center. This new service provides U.S. **Department of Defense** (**DoD**) customers with the ability to deploy compute and storage ...

미국 DoD

네이버 클라우드 플랫폼 공공기관용
https://www.gov-ncloud.com › intro › publicCloud

행정·공공기관 클라우드 전환 사업

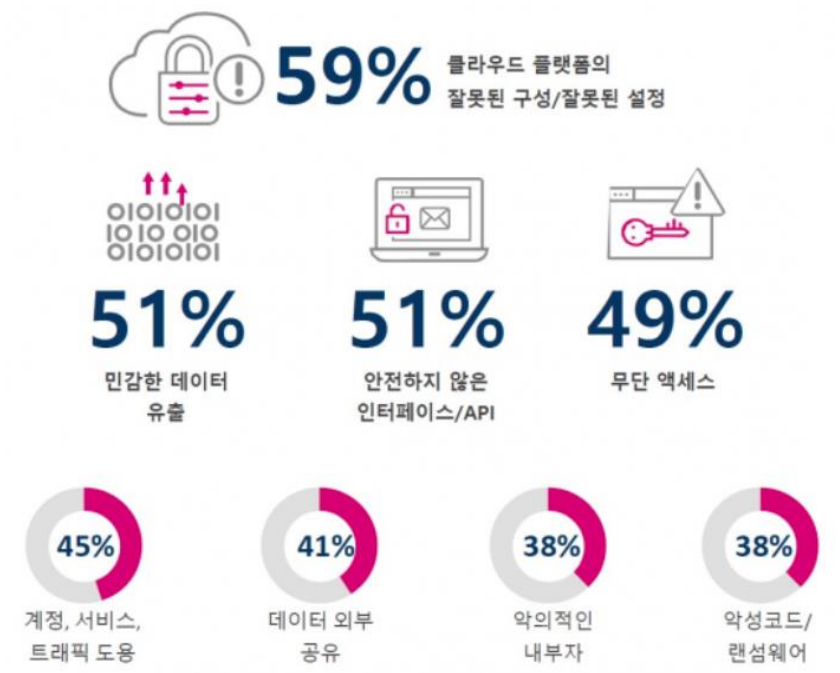
네이버 클라우드 플랫폼이 최적입니다. ... '코로나19 백신예방접종 사전 예약 시스템'에 ... 네이버클라우드의 원격 관리 서비스를 받을 수 있는 하이브리드 클라우드입니다.

국내 정부기관

국내외 기업은 물론 정부 기관에서도 클라우드 네이티브 환경으로의 전환이 빠르게 진행되고 있음

클라우드 네이티브 환경에서의 보안 위협

클라우드 환경에서 가장 많이 발생하는 사고유형



출처: 체크포인트 2023 클라우드 보안 보고서

주요 보안 사고 원인

- 잘못된 구성/잘못된 설정 (Misconfiguration)
  - 권한 및 보안 그룹을 잘못 설정
  - 복잡한 보안 설정을 놓치거나 오류 발생
  - **암호화가 적용되지 않은 민감 데이터 노출**
- 민감한 데이터 유출 (Sensitive Data Breach)
  - 안전하지 않은 네트워크 구성으로 인해 발생
  - 무단 접근이나 보안 취약점 악용
  - **암호화가 적용되지 않은 민감 데이터 노출**
- 안전하지 않은 인터페이스/API (Insecure Interface/API)
  - API 및 관리 인터페이스의 보안 취약점 악용
  - 인증 및 접근 제어 메커니즘의 부족
  - **데이터 암호화와 통신 보안의 부실한 상태에서 외부 공격 노출**

국내외 클라우드 네이티브 환경으로의 활발한 전환 가운데 다수의 보안 위협도 함께 증가!
그 중에서도 많은 부분이 부실한 암호화 체계 관리로 인하여 발생!

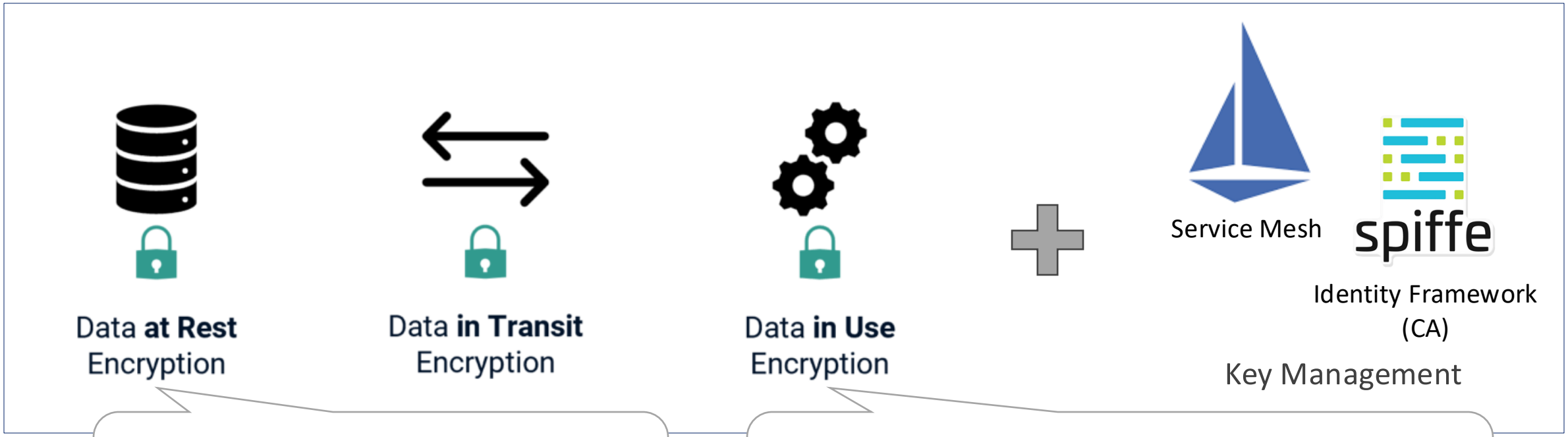
암호화 체계의 필요성

- **주요 보안 사고 요인에 대한 대응**
  - 잘못된 구성/잘못된 설정
    - 데이터가 안전하게 보호되고 있는지 실시간으로 암호화 상태 모니터링 필요
  - 민감한 데이터 유출
    - 데이터가 적절히 암호화되었는지, 암호화 키가 안전하게 관리되고 있는지 모니터링 필요
  - 안전하지 않은 인터페이스/API
    - 데이터 통신 시 데이터가 안전하게 암호화되었는지 모니터링 필요

- **실시간 암호화 상태 모니터링과 암호화 키 관리의 투명성 확보**
  - 암호화 체계는 클라우드 환경에서 발생하는 주요 보안 사고를 사전에 예방을 위해 필요
  - 보안의 가시성을 높이고 신뢰성 있는 클라우드 환경을 구축하기 위해 필요

클라우드 환경에서의 데이터 암호화

- **데이터 암호화**
  - 클라우드 환경에서 데이터를 보호하기 위한 가장 기본적이면서 강력한 수단

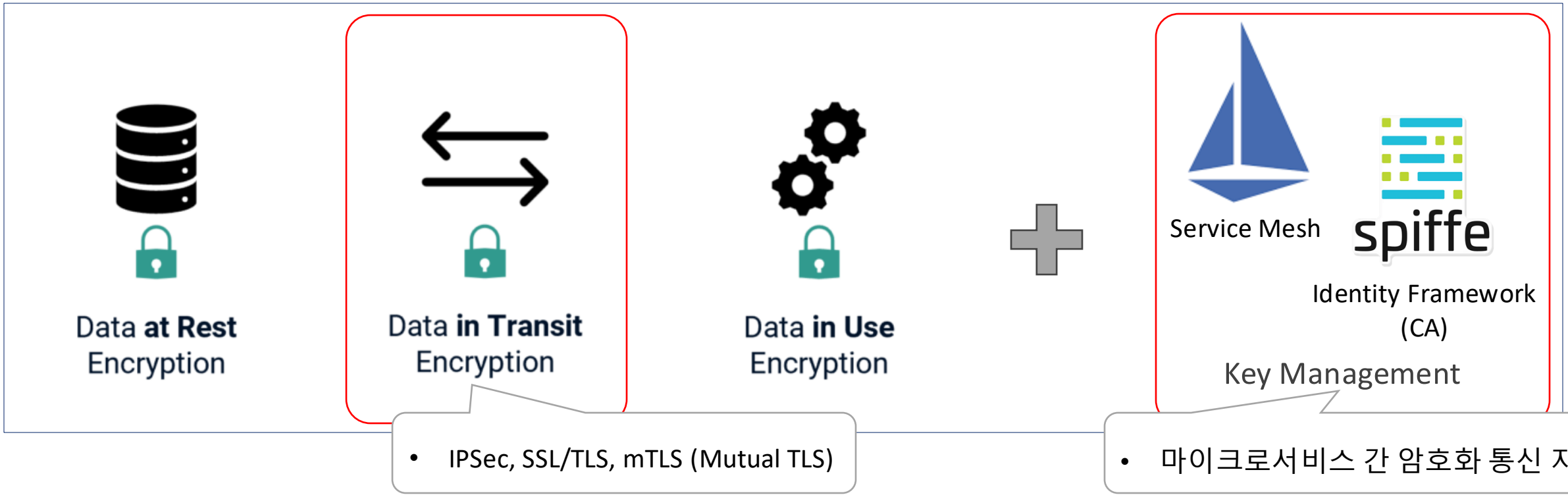


- javax.crypto, openssl/rsa.h
- **애플리케이션 코드에 대한 접근 필요**

- Trusted Execution Environment (TEE) – Intel SGX, AMD SEV
- **별도의 특정 하드웨어 필요**

클라우드 환경에서의 데이터 암호화

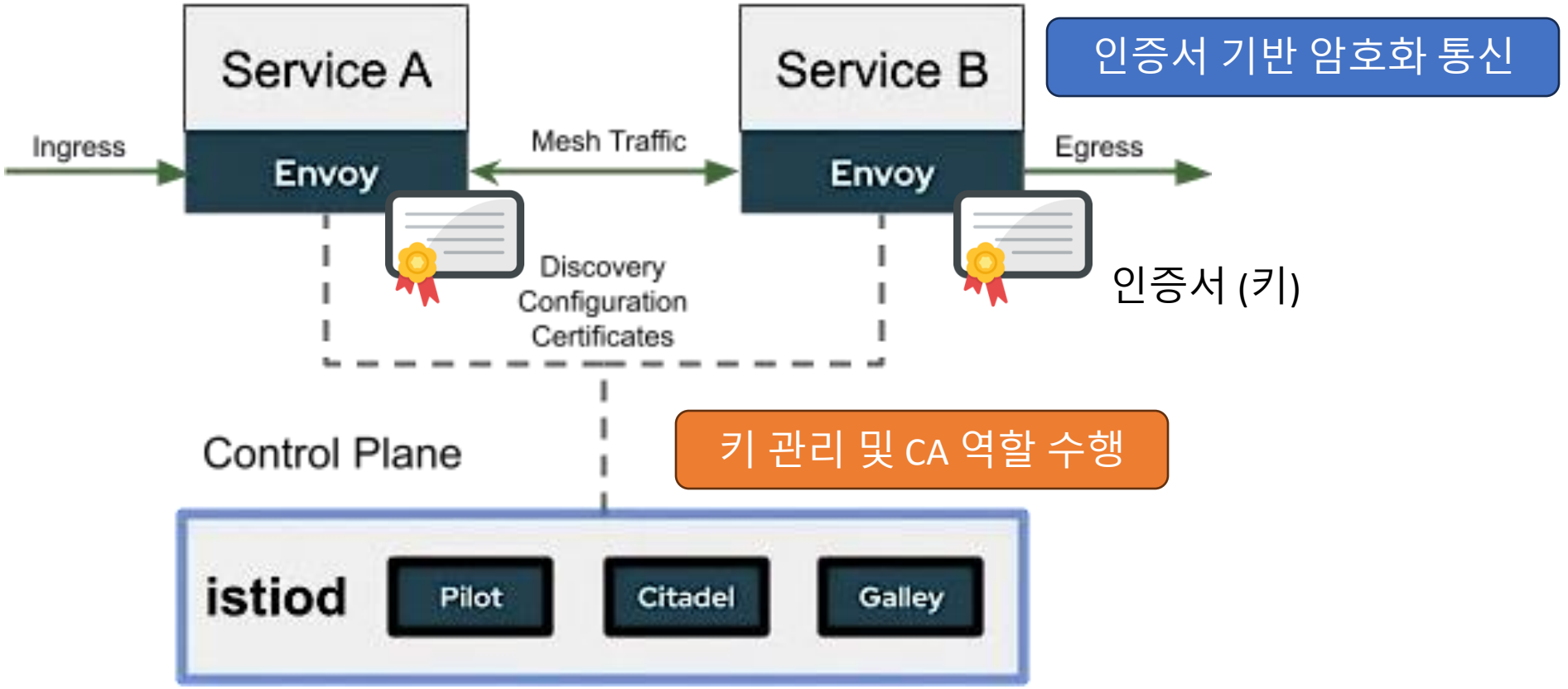
- **데이터 암호화**
  - 클라우드 환경에서 데이터를 보호하기 위한 가장 기본적이면서 강력한 수단



애플리케이션 또는 환경의 변경 없이 적용 가능한 Data in Transit 부분과 Key Management 부분에 초점을 맞춤

클라우드 환경에서의 키 관리 시스템

- **클라우드 환경에서의 키 관리 체계**



출처: https://www.redhat.com/en/blog/introducing-openshift-service-mesh-2.0

클라우드 환경에서의 키 관리 시스템

- **키 관리 시스템(KMS)**
  - 클라우드 환경에서 데이터 통신 보안을 위해 중요한 암호화 키를 생성, 저장, 배포 및 관리하는 역할을 수행

- **주요 기능**
  - 키 생성 및 배포
    - 각 서비스 간의 암호화 통신을 보장하기 위해 필요한 암호화 키를 생성하고, 각 서비스 또는 프록시로 적절히 배포
  - 키 롤링 (Key Rolling)
    - 주기적으로 암호화 키를 자동으로 교체
  - 키 폐기
    - 사용하지 않거나 만료된 키를 안전하게 폐기하여 오래된 키로 인한 보안 위협을 최소화
  - 중앙 집중화된 관리
    - 암호화 키를 중앙에서 관리함으로써, 각 서비스가 키를 관리하는 복잡성을 줄이고 보안성 강화
  - 루트 인증서 및 중간 인증서
    - 중앙에서 관리되는 루트 인증 기관(CA)이 각 서비스의 인증서를 발급하고 관리

클라우드 환경에서의 키 관리 시스템

- **(각 서비스가 가지고 있는) 인증서**
  - 서비스 간 통신을 인증하고 보호하는 중요한 수단
  - 특히, mTLS와 같은 암호화 통신 프로토콜에서 핵심적인 역할을 수행
    - mTLS → 각 서비스는 서로를 신뢰하기 위해 인증서를 통해 자신이 신뢰할 수 있는 엔티티임을 증명한 후 암호화 통신

- **주요 기능**
  - 서비스 인증
    - 각 서비스는 통신 전 자신의 신원을 증명하기 위해 인증서를 사용하여 상호 인증 수행
  - 데이터 암호화
    - 주어진 인증서를 기반으로 통신할 때 데이터를 암호화
  - 자동 갱신
    - 클라우드 환경 내 인증서는 일반적으로 짧은 유효 기간을 갖기 때문에, 자동 갱신 메커니즘이 필수

클라우드 환경에서의 키 관리 체계 가시화

- **클라우드 환경에서의 전반적인 인증서 사용 흐름을 모니터링**
  - 실시간으로 모든 서비스들에 대한 인증서 현황 파악

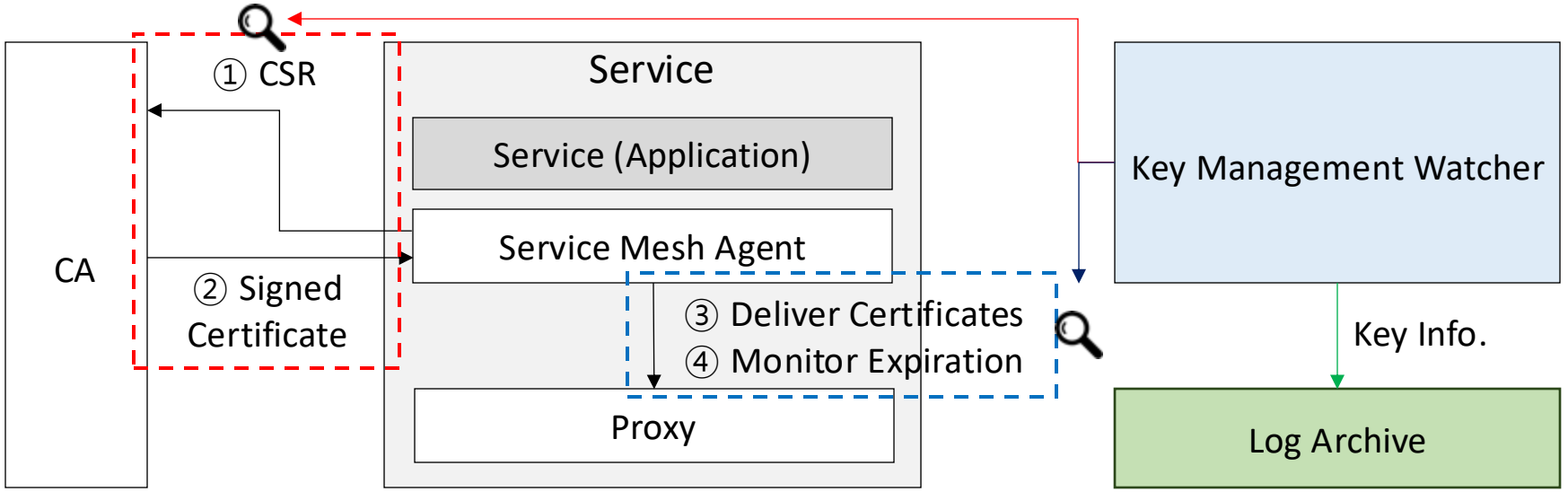
- **실시간으로 인증 기관(CA)의 활동을 추적하여 인증서 발급, 갱신, 폐기 과정을 모니터링**
  - CA 로그 분석을 통한 각 서비스로 발급되는 인증서에 대한 라이프사이클 추적

- **인증서의 수명 주기를 추적하여 만료 예정 인증서를 식별, 자동 갱신 프로세스 모니터링**
  - 만료 예정 인증서에 대한 자동 갱신 프로세스를 추적하고, 기존 인증서에 대한 맥락 유지

클라우드 환경에서의 키 관리 체계 가시화 구현

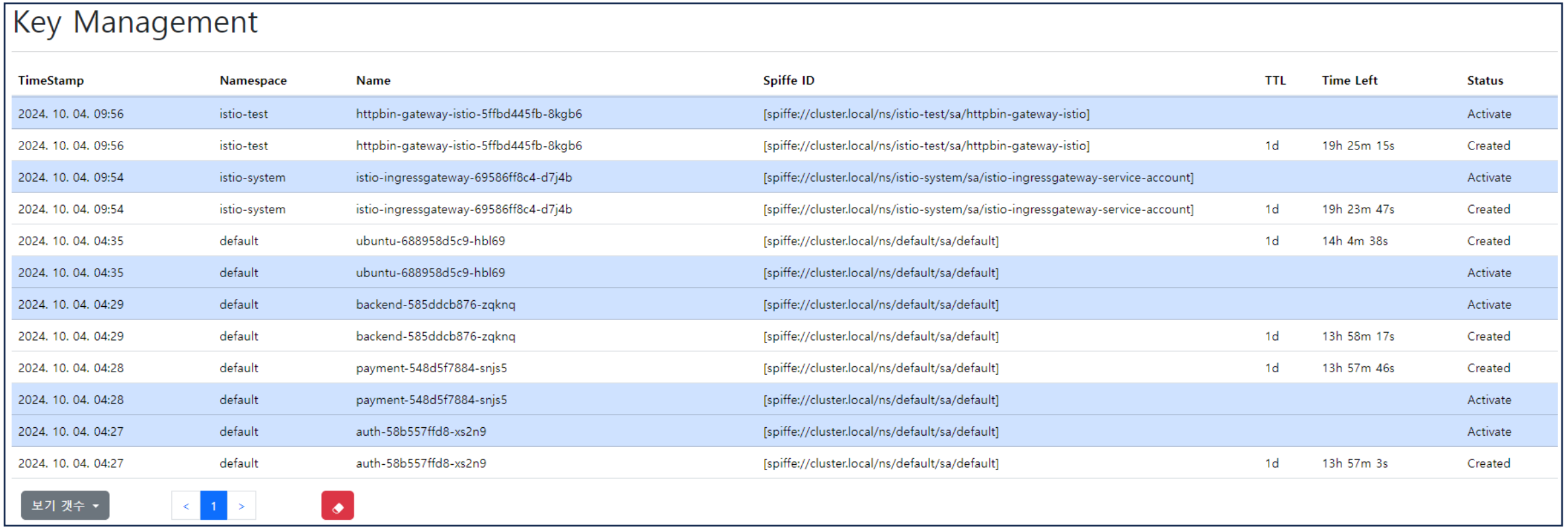
- **키 관리 시스템과 키 사용 주체에 대한 모니터링**
  - 컨트롤 플레인(키 관리 시스템)의 CA 로그를 실시간으로 수집 및 분석
  - 키 사용 주체(각 서비스에 대한 프록시)로부터 키 사용에 대한 Metrics 정보를 실시간으로 수집 및 분석
  - 각각의 소스로부터 수집된 정보를 결합하여 종합적인 인증서 발급/사용/폐기 등의 상태 파악

- **가시화를 위해 수집된 정보를 중앙의 데이터베이스에 저장**
  - 기록 보관 및 빠른 검색을 위한 효율적인 Log Archiving System 구현



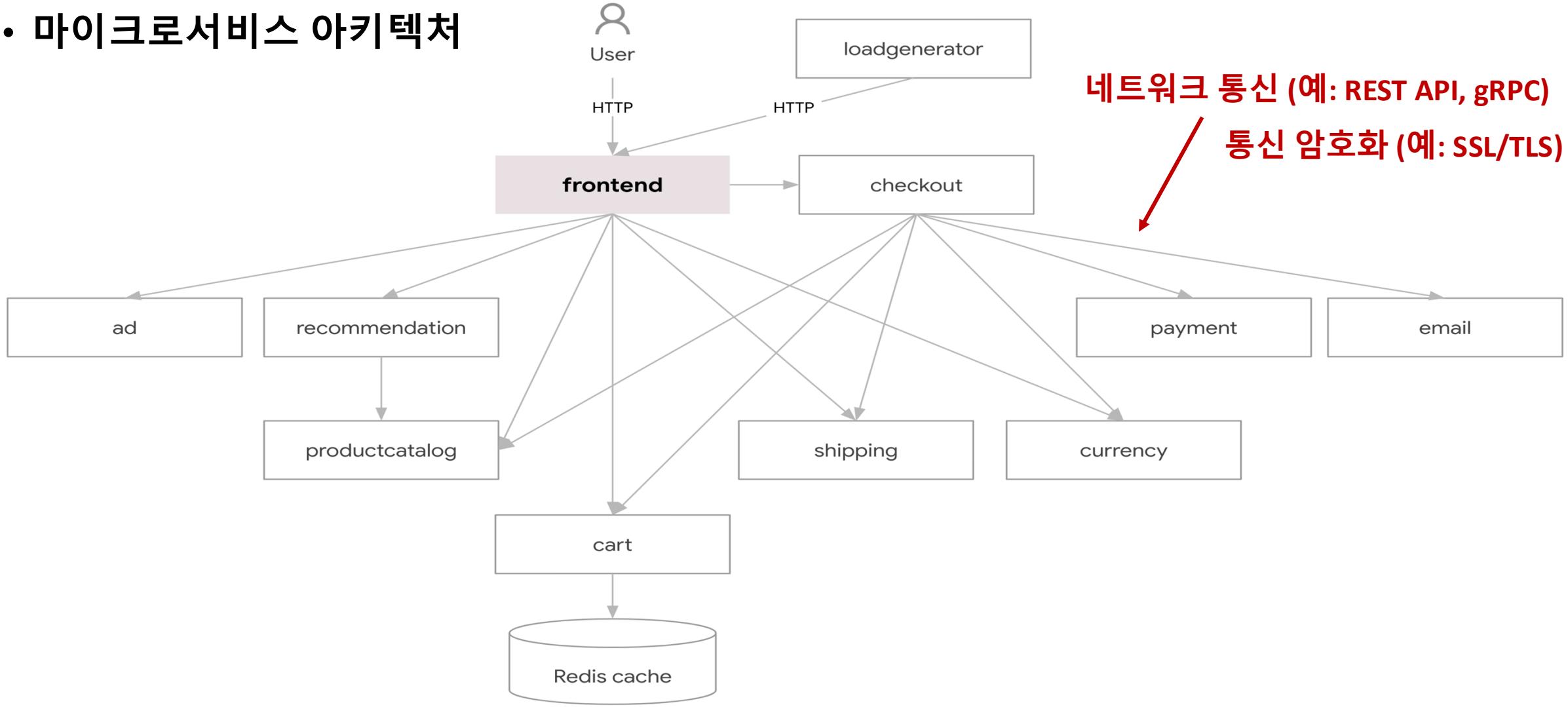
클라우드 환경에서의 키 관리 체계 가시화 구현

- **키 관리 체계 모니터링 대시보드**
  - 인증서 생성, 활성화, 갱신, 만료 등 전반적인 라이프사이클을 한 눈에 확인할 수 있음
  - 인증서의 유효 시간은 실시간으로 업데이트



클라우드 환경에서의 암호화 통신

- 마이크로서비스 아키텍처



출처: https://github.com/GoogleCloudPlatform/microservices-demo

클라우드 환경에서의 암호화 통신 가시화

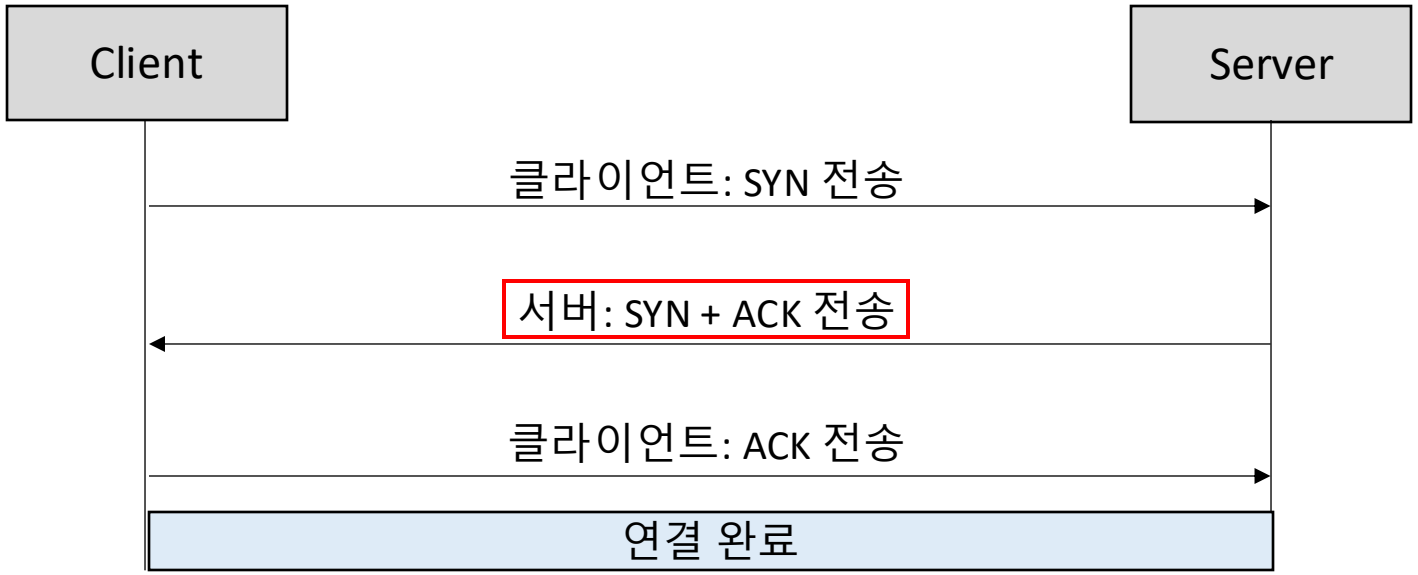
- **세션 추적**
  - 모든 네트워크 세션 추적
    - 클라우드 환경 내 **모든 서비스**에서 발생하는 **네트워크 세션의 생성, 유지, 종료**까지 추적
  - 암호화되지 않은 세션 식별
    - 암호화 통신에 대한 가시화를 목적으로 하지만, **암호화 되지 않은 세션** 역시 반드시 식별할 필요가 있음
  - 고성능 네트워크 모니터링
    - 수많은 서비스들이 발생 시키는 네트워크 통신을 **손실 없이 효율적으로 모니터링**할 수 있어야 함

- **암호화 알고리즘 협상 과정 추적**
  - 암호화 세션 감시
    - 암호화된 세션을 식별한 후, 해당 세션의 **암호화 과정과 관련된 정보**를 실시간으로 추적
  - 암호화 요소 수집
    - 세션에서 사용되는 다양한 암호화 요소(**암호화 알고리즘, 인증서 서명 알고리즘, 키 교환 서명 알고리즘 등**) 수집
  - 암호화 알고리즘 취약성 분석
    - 암호화된 세션도 사용된 암호화 알고리즘이 취약할 수 있는 만큼, 사용 중인 **알고리즘에 대한 취약성 검사**

세션 추적 – 네트워크 세션의 시작

- **TCP 3-Way Handshake**
  1. 클라이언트가 SYN 플래그가 설정된 패킷을 서버로 전송
  2. 이를 수신한 서버가 SYN, ACK플래그가 설정된 패킷으로 응답
  3. 클라이언트는 SYN-ACK 패킷을 수신한 후 ACK 플래그가 설정된 패킷을 전송

  - **2단계의 SYN, ACK 플래그가 설정된 패킷을 감지하여 세션 시작을 추적**



세션 추적 – 네트워크 세션의 끝

- **TCP 4-Way Handshake**
  1. 연결을 종료하고자 하는 쪽에서 FIN 플래그가 설정된 패킷을 전송
  2. 수신 측에서 ACK 플래그가 설정된 패킷으로 응답
  3. 수신 측에서 남은 데이터 전송을 모두 마친 후 FIN 플래그가 설정된 패킷을 전송
  4. 종료하고자 하는 측에서 ACK 패킷을 전송

  - **2, 3단계의 ACK, FIN 플래그가 설정된 패킷을 감지하여 세션 종료를 추적**

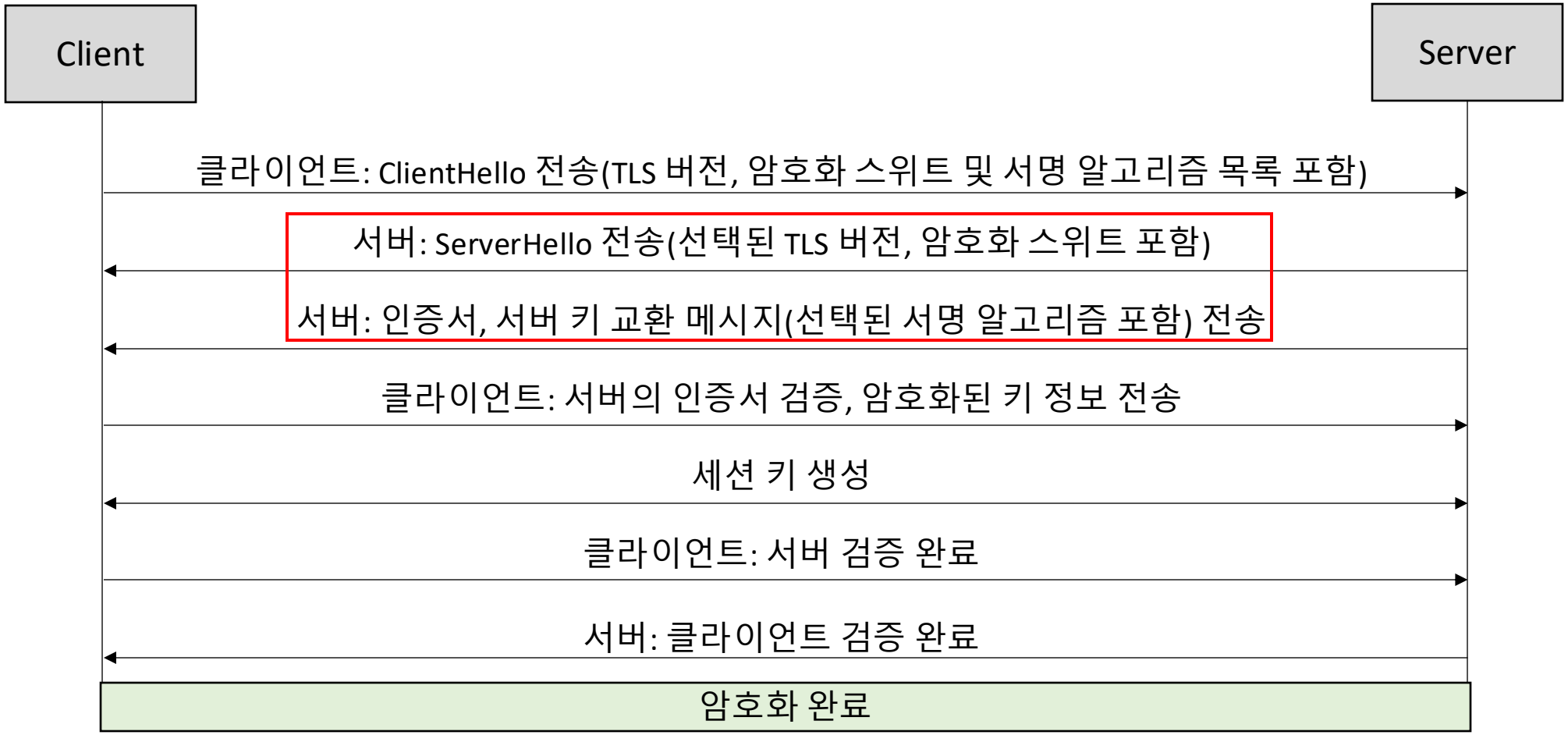
암호화 알고리즘 협상 과정 추적

- **Transport Layer Security (TLS) 프로토콜**
  - TLS Handshake라는 협상 과정을 거쳐서 통신 암호화
  - 이 과정에서 **TLS 프로토콜 버전, 암호화, 인증서 서명, 키 교환 알고리즘 등**의 중요한 정보가 교환됨

- **암호화 체계 가시화를 위해 필요한 요소**
  - **암호화 프로토콜 버전**
    - 사용 중인 암호화 프로토콜 버전을 모니터링하여 보안 표준 및 규정 준수 여부를 파악
  - **암호화 스위트**
    - 키 교환, 인증, 암호화, 메시지 무결성 검증 등에 사용되는 암호화 알고리즘 조합의 현황과 취약점을 파악
  - **인증서 서명 알고리즘**
    - 디지털 인증서 전자 서명에 사용된 알고리즘 종류와 안전성을 확인하고 분석
  - **키 교환 알고리즘 및 타원 곡선 암호(ECC)**
    - 클라이언트와 서버 키 교환에 사용되는 알고리즘과 타원 곡선 암호의 안전성을 평가

TLS Handshake 과정

- **TLS Handshake 과정 중 특히 ServerHello에 집중**



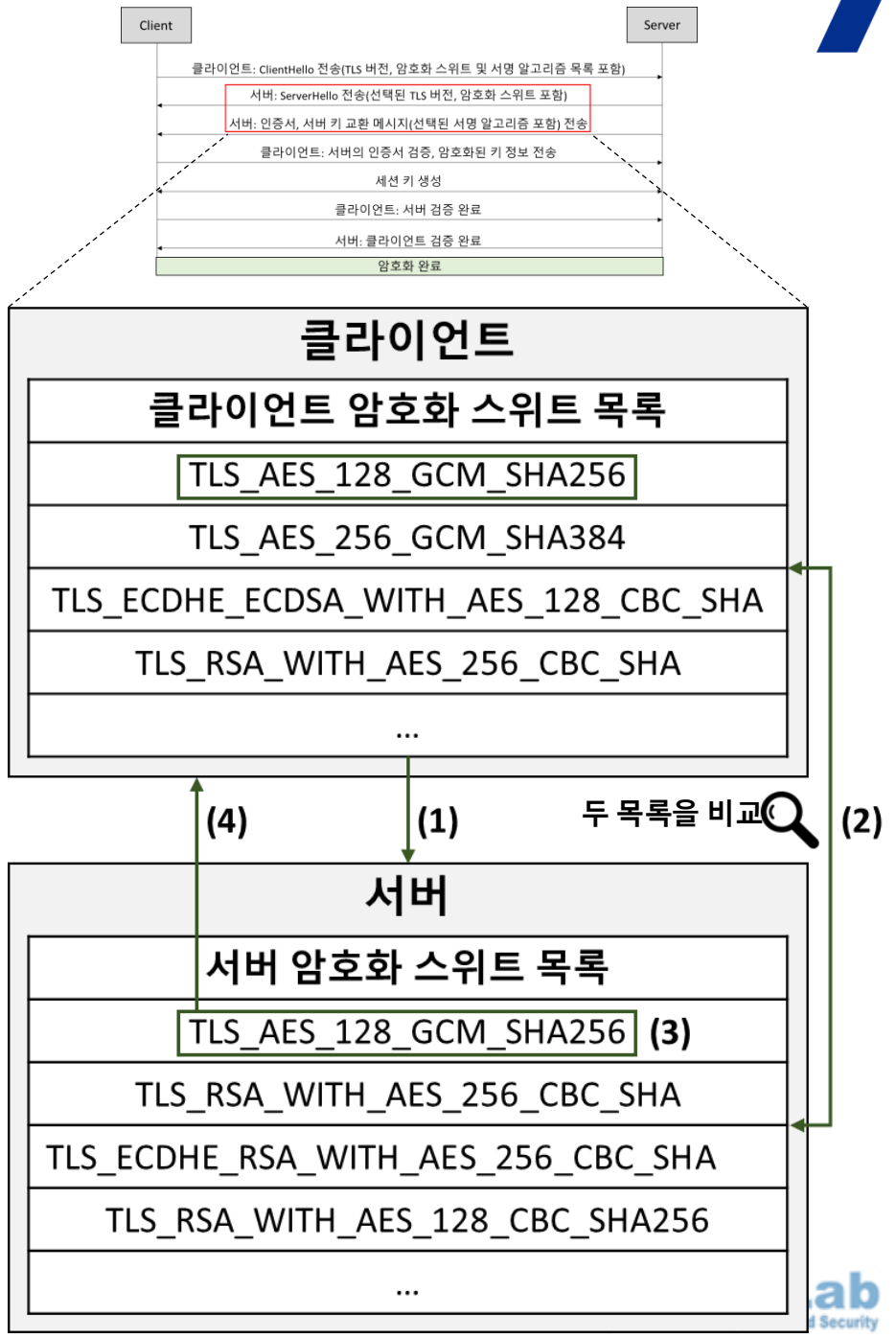
Server Hello에 집중해야 하는 이유

1. 클라이언트가 서버로 암호화 통신 요청 **(Client Hello)**
   - 클라이언트는 사용 가능한 모든 암호화 스위트 목록을 함께 제공

2. 클라이언트로부터 전달된 암호화 스위트 목록을 확인, 서버가 제공 가능한 암호화 스위트 목록과 비교

3. 목록 중 사용 가능하며 가장 안전한 암호화 스위트를 선택

4. 선택된 암호화 스위트를 클라이언트로 회신 **(Server Hello)**
   - **최종적으로 사용할 암호화 스위트는 서버에 의해서 결정**



TLS 버전에 따른 식별 가능 정보

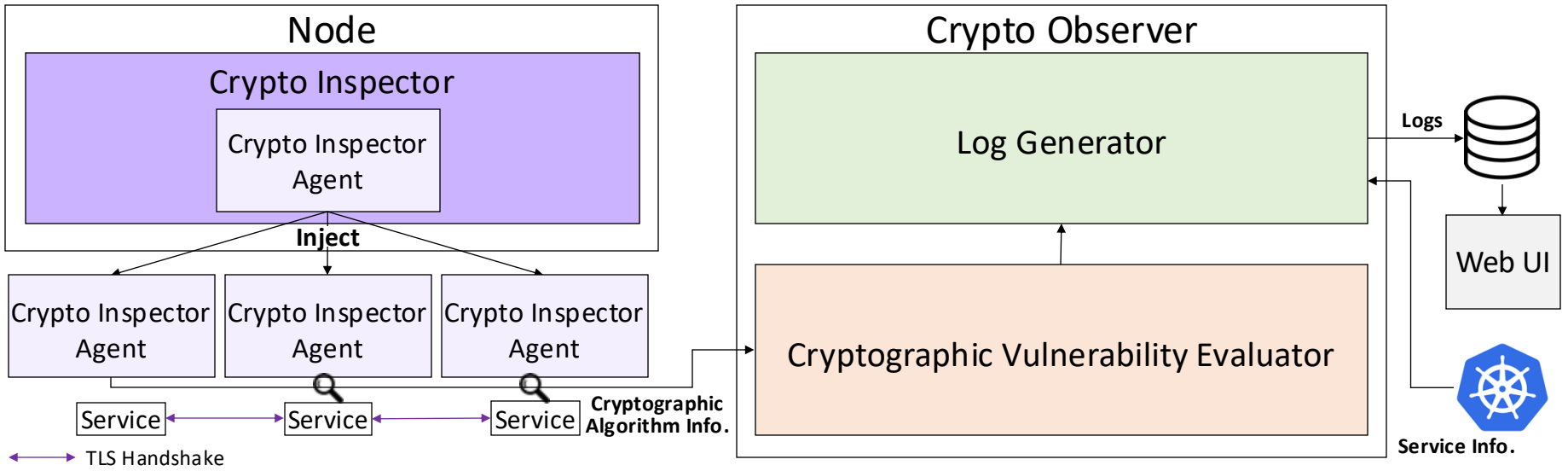
- **TLS Handshake 협상 내용 확인**
  - Handshake 합의 내용은 평문으로 전달

- **TLS 버전에 따른 식별 가능한 협상 메시지와 세부 정보**
  - TLS 1.2까지는 모든 협상 내용을 확인할 수 있고, 1.3부터는 일부 협상 내용만 확인할 수 있음

버전	Server Hello		Certificates	Server Key Exchange	
	TLS 버전	암호화 스위트	인증서 서명	키 교환	타원 곡선 암호
~ TLS 1.2	O	O	O	O	O
TLS 1.3	O	O	X	X	X

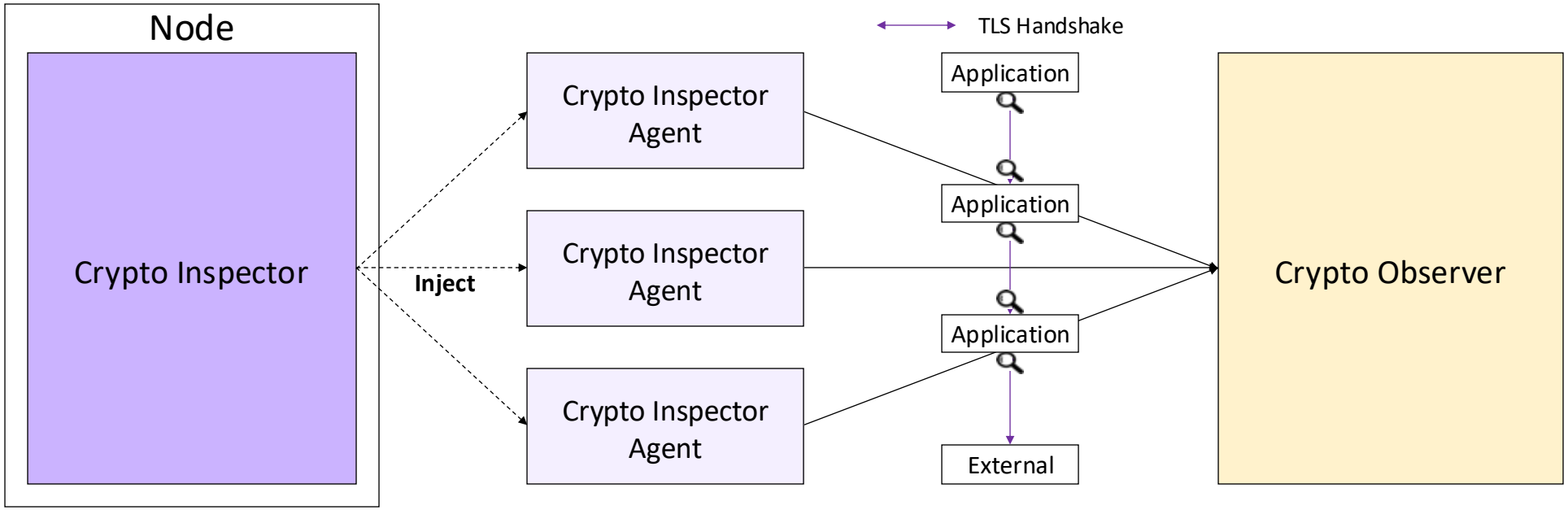
클라우드 환경에서의 암호화 통신 가시화 구현

- **전체 시스템 구조**
  - Crypto Inspector – 노드 별로 각 서비스의 통신 과정에서 사용되는 암호화 알고리즘을 지속적으로 수집
    - Crypto Inspector Agent – 각 서비스로부터 실질적인 네트워크 통신 암호화 과정을 모니터링
  - Crypto Observer – 전체 시스템 내 중앙집중적으로 암호화 관련 정보를 수집하고 관리
    - 각각의 Crypto Inspector Agent로부터 수집된 암호화 정보에 대해서 취약성 평가
    - 각 정보의 소스인 서비스에 대한 사용자 메타데이터 (예: 서비스 이름)을 수집된 암호화 정보와 통합 후 DB에 저장
    - 이후 저장된 정보는 Web UI를 통해 가시화됨



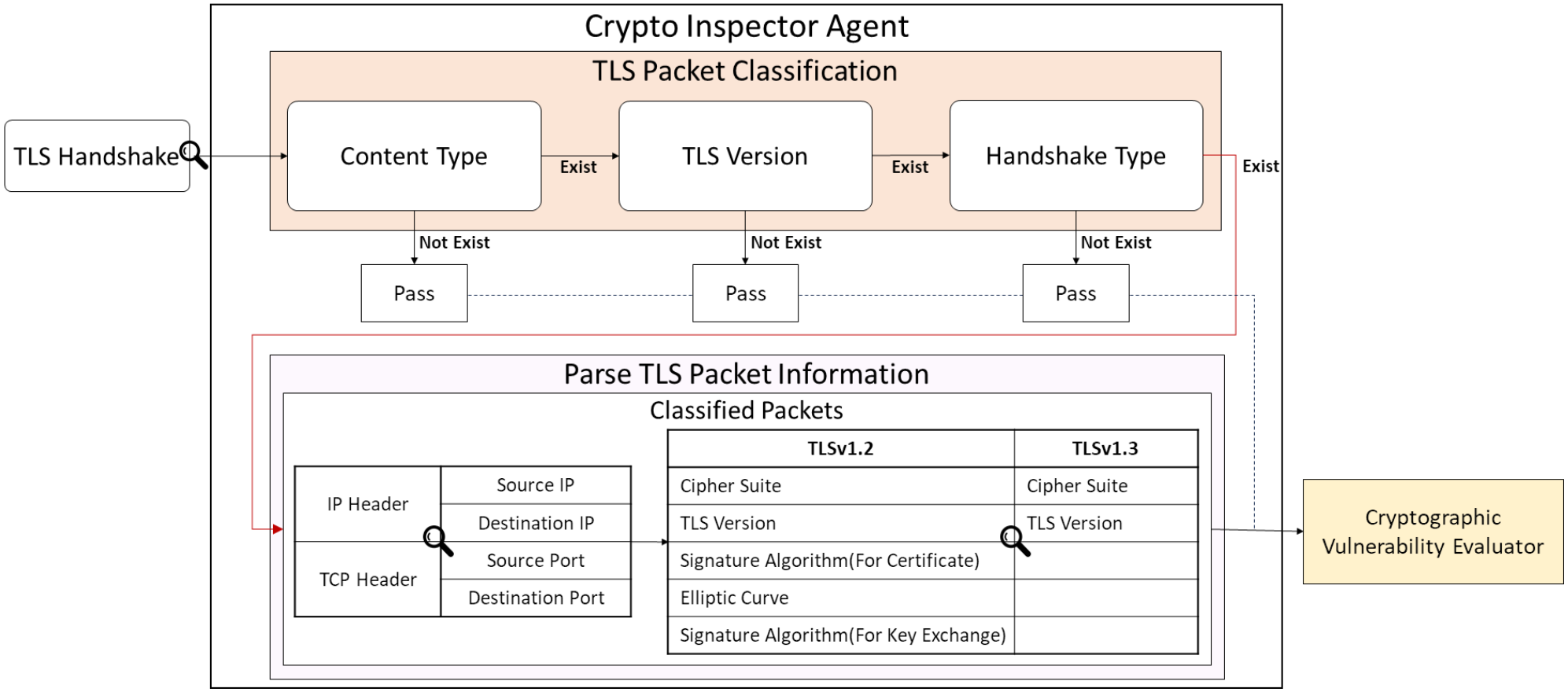
클라우드 환경에서의 암호화 통신 가시화 구현

- **Crypto Inspector**
  - 각 노드에서 동작 중인 서비스(애플리케이션)를 실시간으로 탐지
  - 새로운 서비스가 생성되면 해당 서비스에 Crypto Inspector Agent를 붙이고, 제거되면 자동으로 Agent도 제거
  - Crypto Inspector Agent는 각 서비스에서 발생하는 모든 네트워크 통신을 모니터링



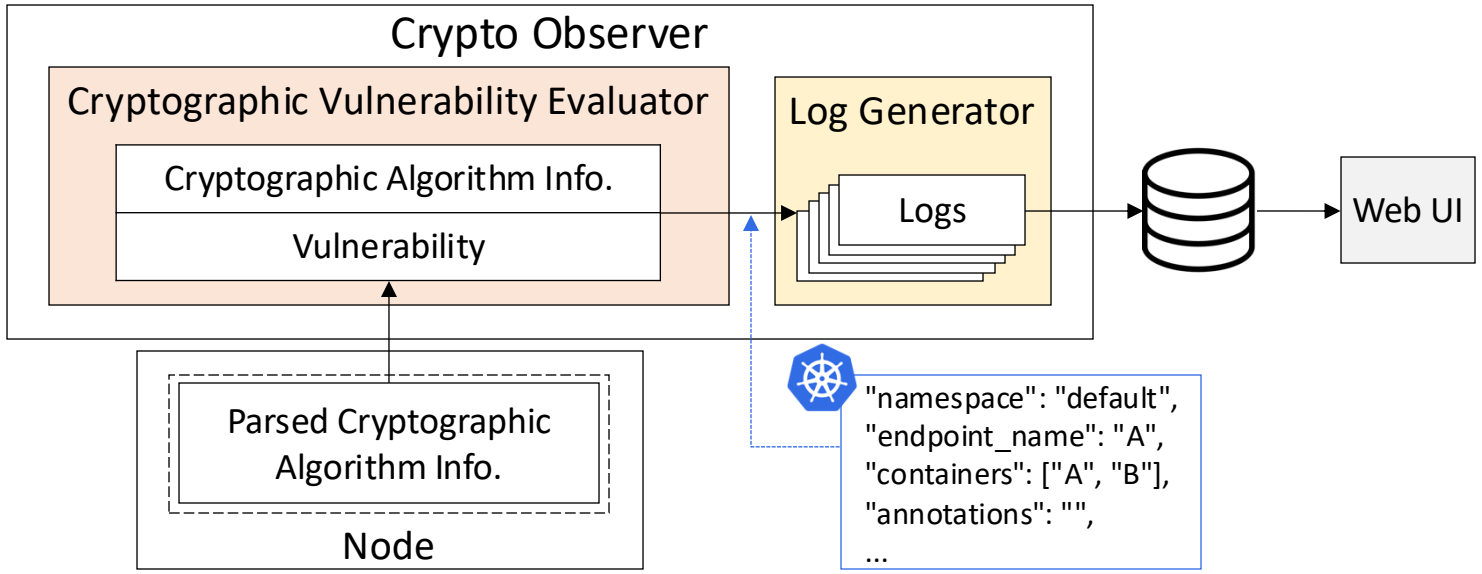
클라우드 환경에서의 암호화 통신 가시화 구현

- **Crypto Inspector Agent**
  - TLS Handshake 과정 추적 및 협상 내용 분석



클라우드 환경에서의 암호화 통신 가시화 구현

- **Crypto Observer**
  - Crypto Inspector로부터 전송 받은 암호화 요소 정보를 분석, 각 세션의 암호화 알고리즘 취약성을 평가
  - IP 주소와 같은 정보를 사용자 메타데이터 (Namespace, Service Name, Labels, Annotations 등)과 함께 통합하여 평가 내용을 로그 형태로 작성한 후 데이터베이스에 저장
  - 데이터베이스에 저장된 내용은 Web UI에 의해 암호화 통신 가시화 과정에서 사용



취약 암호화 알고리즘의 분류와 정의

- **대표적인 취약 암호화 알고리즘 예시**

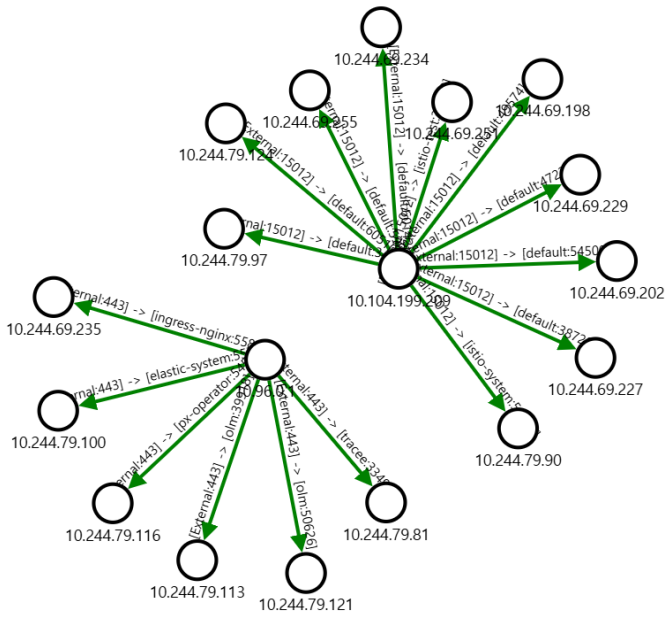
암호화 스위트 구성요소	취약점	잠재적 영향
RC4	편향성 문제로 인해 평문 복구 가능성이 존재	암호화된 데이터의 기밀성 침해, 중요 정보 노출 위험
DES, 3DES	현대의 컴퓨팅 파워로 Brust Force 공격 가능성이 있음	암호화된 데이터의 해독, 기밀 정보 유출, 시스템 성능 저하
MD5, SHA-1	충돌 저항성이 약해 다른 입력으로 같은 해시 발생 가능	디지털 서명 위조, 인증 우회, 데이터 무결성 검증 실패

- **암호화 세션의 취약도 분류 기준**
  - 하나 이상의 암호화 구성 요소가 암호화되지 않은 경우 → 암호화 되지 않음
  - 하나 이상의 암호화 알고리즘이 '취약함'으로 분류될 경우 → 취약한 암호화
  - 모든 요소가 안전할 경우 → 안전한 암호화

클라우드 환경에서의 암호화 통신 가시화 구현

- **취약 암호 탐지 – 토폴로지 뷰 (안전한 암호화 예시)**
  - 서비스 간의 통신을 토폴로지 형태로 표현, 각 통신에 대한 취약 정도를 색으로 표시
    - 초록색 간선과 초록색 테이블은 안전한 세션임을 의미

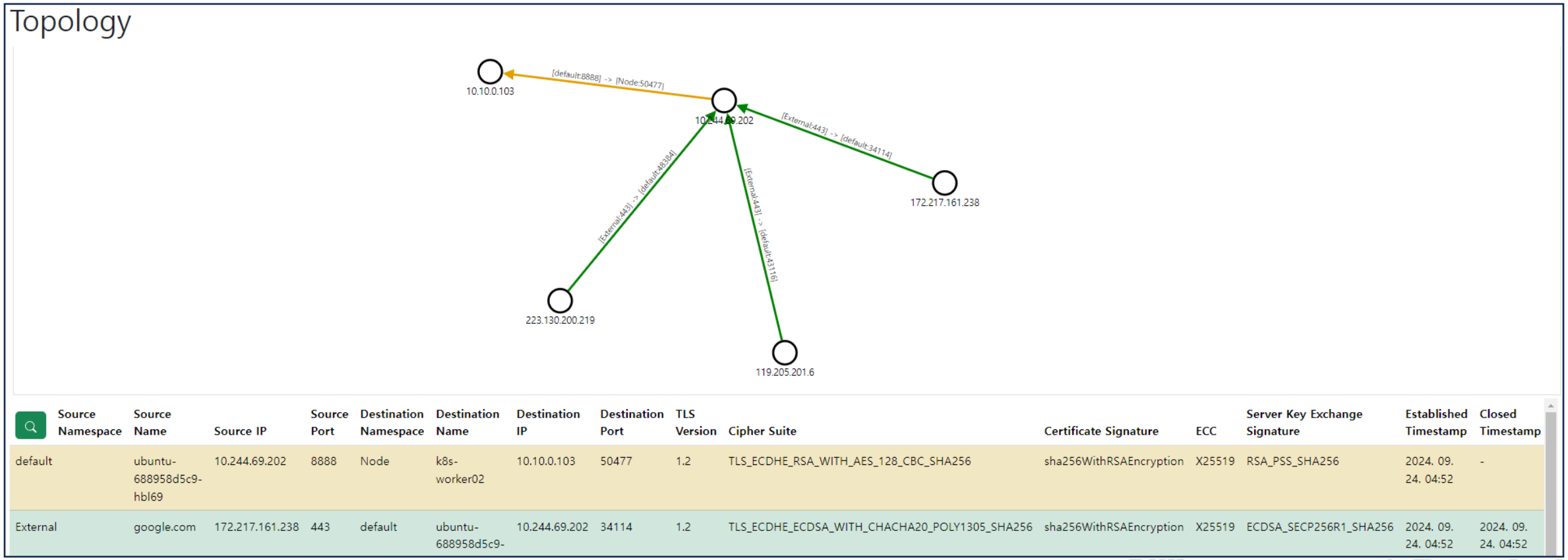
Topology



Source Namespace	Source Name	Source IP	Source Port	Destination Namespace	Destination Name	Destination IP	Destination Port	TLS Version	Cipher Suite	Certificate Signature	ECC	Key Exchange Signature	Established Timestamp	Closed Timestamp
External	istiod.istio-system.svc.cluster.local	10.104.199.209	15012	default	nginx	10.244.79.97	34928	1.3	TLS_AES_128_GCM_SHA256	-	-	-	2024. 09. 20. 17:01	2024. 09. 20. 03:28

클라우드 환경에서의 암호화 통신 가시화 구현

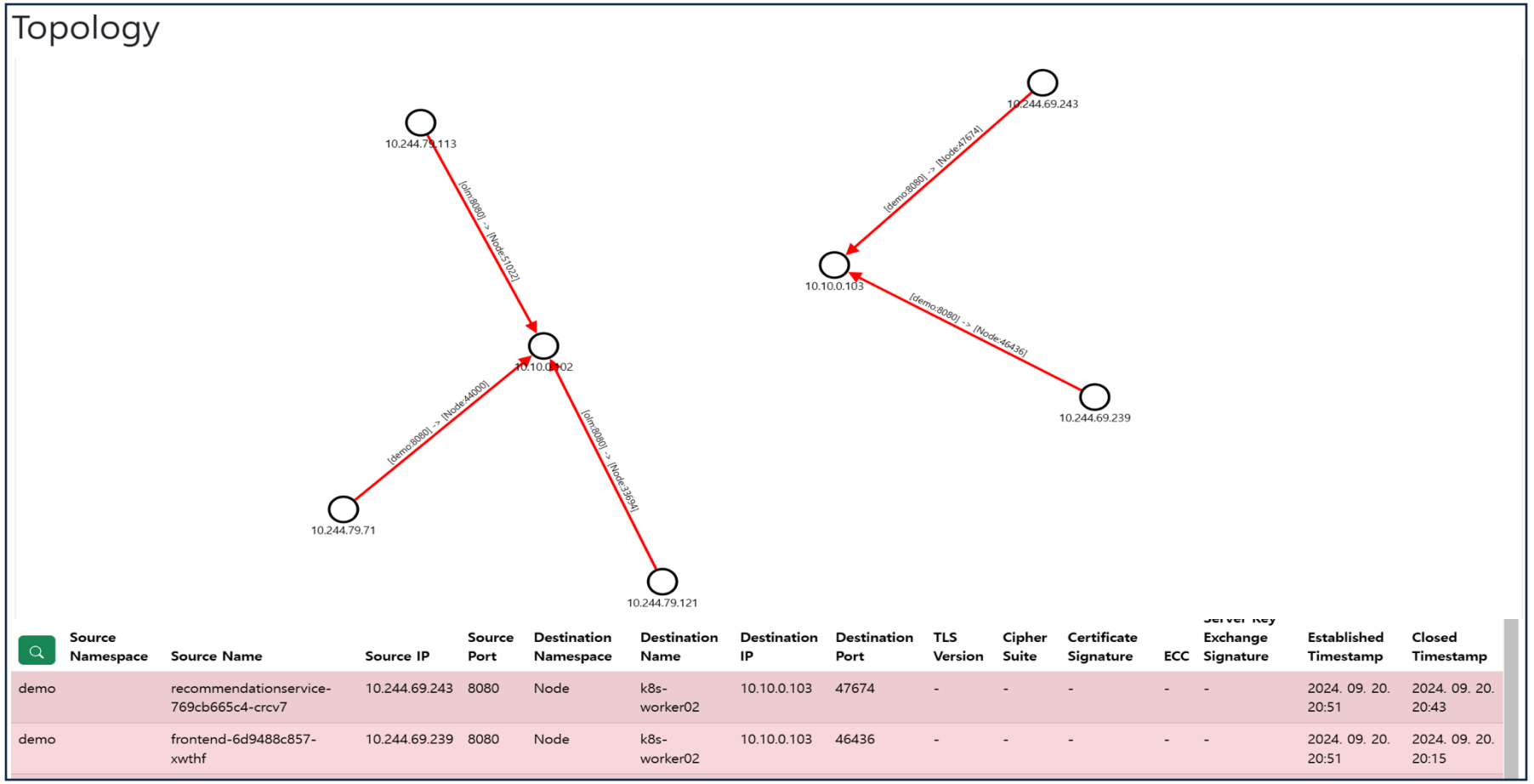
- **취약 암호 탐지 – 취약한 암호화 예시**
  - 취약한 암호화의 경우, 아래 그림과 같이 노란색 간선과 노란색 테이블로 표시
    - 토폴로지 그래프의 간선 혹은 하단 테이블 선택 시 세부내용 확인 가능



Source Namespace	Source Name	Source IP	Source Port	Destination Namespace	Destination Name	Destination IP	Destination Port	TLS Version	Cipher Suite	Certificate Signature	ECC	Server Key Exchange Signature	Established Timestamp	Closed Timestamp
default	ubuntu-688958d5c9-hbl69	10.244.69.202	8888	Node	k8s-worker02	10.10.0.103	50477	1.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	sha256WithRSAEncryption	X25519	RSA_PSS_SHA256	2024. 09. 24. 04:52	-
External	google.com	172.217.161.238	443	default	ubuntu-688958d5c9-	10.244.69.202	34114	1.2	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	sha256WithRSAEncryption	X25519	ECDSA_SECP256R1_SHA256	2024. 09. 24. 04:52	2024. 09. 24. 04:52

클라우드 환경에서의 암호화 통신 가시화 구현

- **취약 암호 탐지 – 암호화되지 않은 세션 예시**
  - 암호화 되지 않은 세션은 빨간색 간선과 테이블로 표시



Source Namespace	Source Name	Source IP	Source Port	Destination Namespace	Destination Name	Destination IP	Destination Port	TLS Version	Cipher Suite	Certificate Signature	ECC	Server Key Exchange Signature	Established Timestamp	Closed Timestamp
demo	recommendationservice-769cb665c4-crcv7	10.244.69.243	8080	Node	k8s-worker02	10.10.0.103	47674	-	-	-	-	-	2024. 09. 20. 20:51	2024. 09. 20. 20:43
demo	frontend-6d9488c857-xwthf	10.244.69.239	8080	Node	k8s-worker02	10.10.0.103	46436	-	-	-	-	-	2024. 09. 20. 20:51	2024. 09. 20. 20:15

클라우드 환경에서의 암호화 통신 가시화 구현

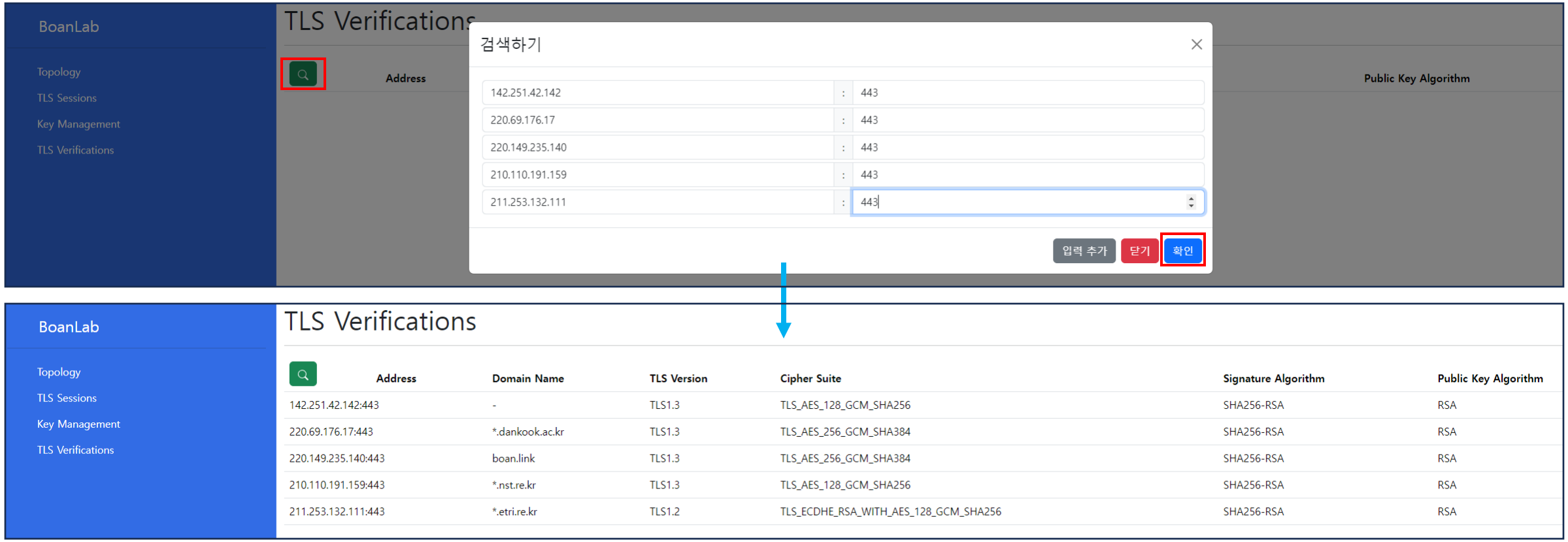
- **취약 암호 탐지 – 세부 세션 정보**
  - 취약 암호 탐지 내용을 텍스트 형식으로 확인할 수 있는 페이지

TLS Sessions

Source Namespace	Source Name	Source IP	Source Port	Destination Namespace	Destination Name	Destination IP	Destination Port	TLS Version	Cipher Suite	Certificate Signature	ECC	Server Key Exchange Signature	Established Timestamp	Closed Timestamp
istio-system	istiod-d4f88d44f-rk4fz	10.244.69.207	15012	default	ubuntu-688958d5c9-hbl69	10.244.69.202	52602	1.3	TLS_AES_128_GCM_SHA256	-	-	-	2024. 09. 26. 16:35	2024. 09. 26. 17:04
istio-system	istiod-d4f88d44f-rk4fz	10.244.69.207	15012	default	backend-585ddcb876-zqknq	10.244.69.234	46308	1.3	TLS_AES_128_GCM_SHA256	-	-	-	2024. 09. 26. 16:33	2024. 09. 26. 17:00
istio-system	istiod-d4f88d44f-rk4fz	10.244.69.207	15012	default	backend-585ddcb876-zqknq	10.244.69.234	33808	1.3	TLS_AES_128_GCM_SHA256	-	-	-	2024. 09. 26. 16:29	2024. 09. 26. 16:59
istio-system	istiod-d4f88d44f-rk4fz	10.244.69.207	15012	default	payment-548d5f7884-snjs5	10.244.69.227	38156	1.3	TLS_AES_128_GCM_SHA256	-	-	-	2024. 09. 26. 16:28	2024. 09. 26. 16:56
istio-system	istiod-d4f88d44f-rk4fz	10.244.69.207	15012	default	auth-58b557ffd8-xs2n9	10.244.69.229	59128	1.3	TLS_AES_128_GCM_SHA256	-	-	-	2024. 09. 26. 16:27	2024. 09. 26. 16:57
istio-system	istiod-d4f88d44f-rk4fz	10.244.69.207	15012	istio-system	istio-ingressgateway-69586ff8c4-d7j4b	10.244.79.90	47126	1.3	TLS_AES_128_GCM_SHA256	-	-	-	2024. 09. 26. 16:21	2024. 09. 26. 16:49

클라우드 환경에서의 암호화 통신 가시화 구현

- **TLS 검증**
  - 실제 서비스 간의 통신 뿐만 아니라, 사용자가 직접 특정 서버에 대한 암호화 정보 추출 가능



암호화 전환의 필요성

- **암호화 알고리즘의 수명 한계**
  - 기존의 암호화 알고리즘 (AES, RSA 등)은 점차 공격 방법이 발전하며 더 이상 안전하지 않을 수 있음

- **양자 컴퓨팅 대비**
  - 양자 컴퓨팅의 발전으로 인해 현재의 대칭 및 비대칭 암호화 방식이 취약해질 수 있으므로,
    양자 내성 암호화(Post-Quantum Cryptography)로의 전환 필요

- **키 길이 증가 요구**
  - 현재 사용 중인 암호화 알고리즘에서 강력한 보안을 위해 더 긴 암호화 키를 요구하는 경우 암호 전환 필요

- **규제 및 표준 준수**
  - 법적 요구 사항이나 보안 표준이 변경될 경우, 적시에 해당 요구를 충족하는 암호로의 전환 필요

암호화 전환의 필요성

- **암호화 민첩성(Cryptographic Agility)**
  - 시스템이 새로운 암호화 기술로의 전환을 유연하게 수행할 수 있는 능력
  - 보안 위협에 빠르게 대응하고 더 강력한 보안 솔루션으로 업그레이드 할 수 있는 능력을 포함

- **암호화 민첩성 달성을 위한 주요 요구사항**
  - 암호화 알고리즘의 동적 관리
    - 암호화 알고리즘을 **동적으로 변경**할 수 있는 기능 지원 (키 관리 시스템과 통합되어 알고리즘 교체 가능)
  - 정책 기반 암호 관리
    - 특정 대상 또는 **요구사항에 맞춰 암호화 방법을 조정**
  - 키 롤오버 및 로테이션
    - 키의 유효기간, 사용량, 또는 보안 상태에 따라 자동으로 갱신하고 교체
  - 플러그인 구조
    - **다양한 암호화 라이브러리를 플러그인 형태**로 쉽게 추가하거나 제거

현 클라우드 환경에서 암호 전환 시 문제점

- **환경적 관점**
  - 대표적인 서비스 메시 솔루션인 이스티오의 경우 TLS 1.3에 대해서 암호화 스위트 지정이 불가
    - 프록시가 사용하는 암호화 라이브러리 자체에서 특정 암호화 스위트를 사용하도록 설계되었기 때문
  - 즉, 암호화 알고리즘에 대한 유연성이 제한됨

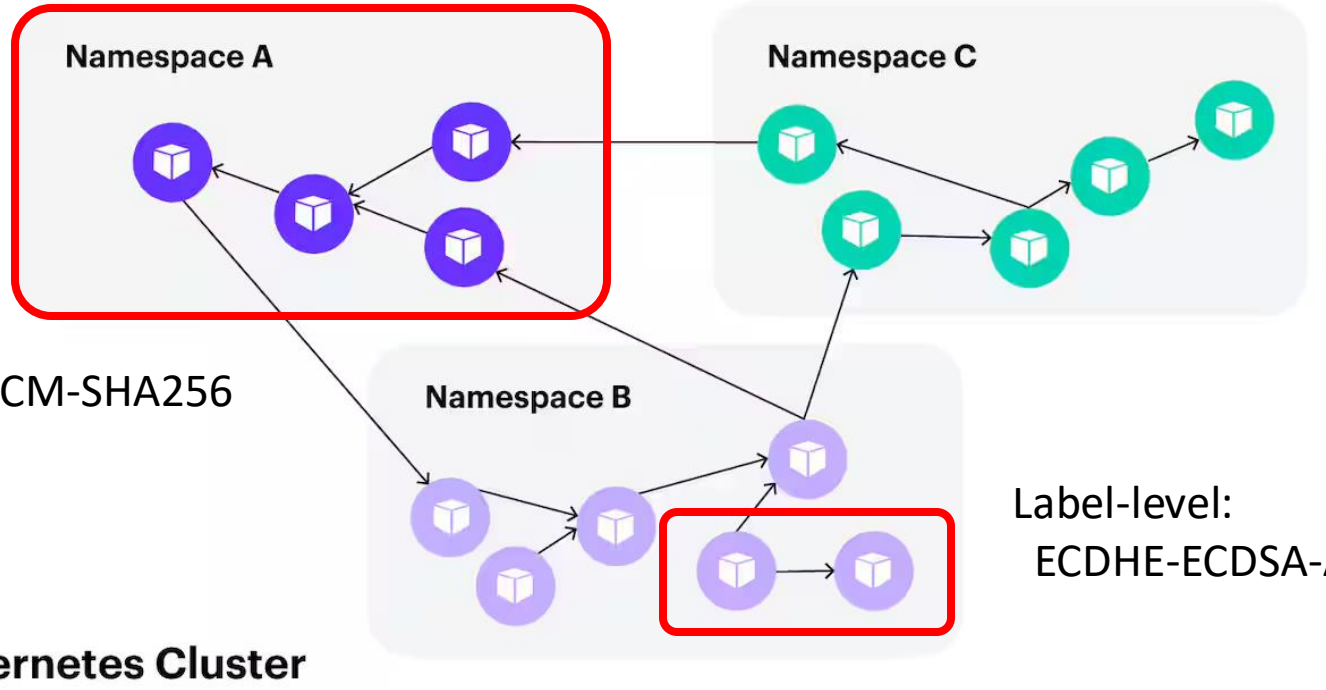
- **기능적 관점**
  - 이스티오의 경우 mTLS를 사용하기 위해서는 클러스터 전체에 동일한 암호화 알고리즘을 설정해야 함
  - 즉, 모든 서비스가 동일한 암호화 알고리즘을 사용해야 함

- **구조적 관점**
  - mTLS 인증서를 발급하기 위해 타원 곡선 알고리즘(ECC)을 사용하는데, 이것이 정적으로 설정되어 있음
  - 즉, 알고리즘들에 대한 동적 선택성과 확장성을 제한됨

암호화 민첩성을 위한 암호 전환 방안 #1

- **암호화 알고리즘 동적 적용**
  - 암호화 정책 설정을 통한 인증서 서명 알고리즘의 유연한 선택 가능
  - Cluster, Namespace, Label 기준으로 다양한 암호화 알고리즘 지정
  - mTLS 통신 시 상황에 따라 암호화 알고리즘 동적 적용

Cluster-level:
ECDHE-RSA-AES256-GCM-SHA384



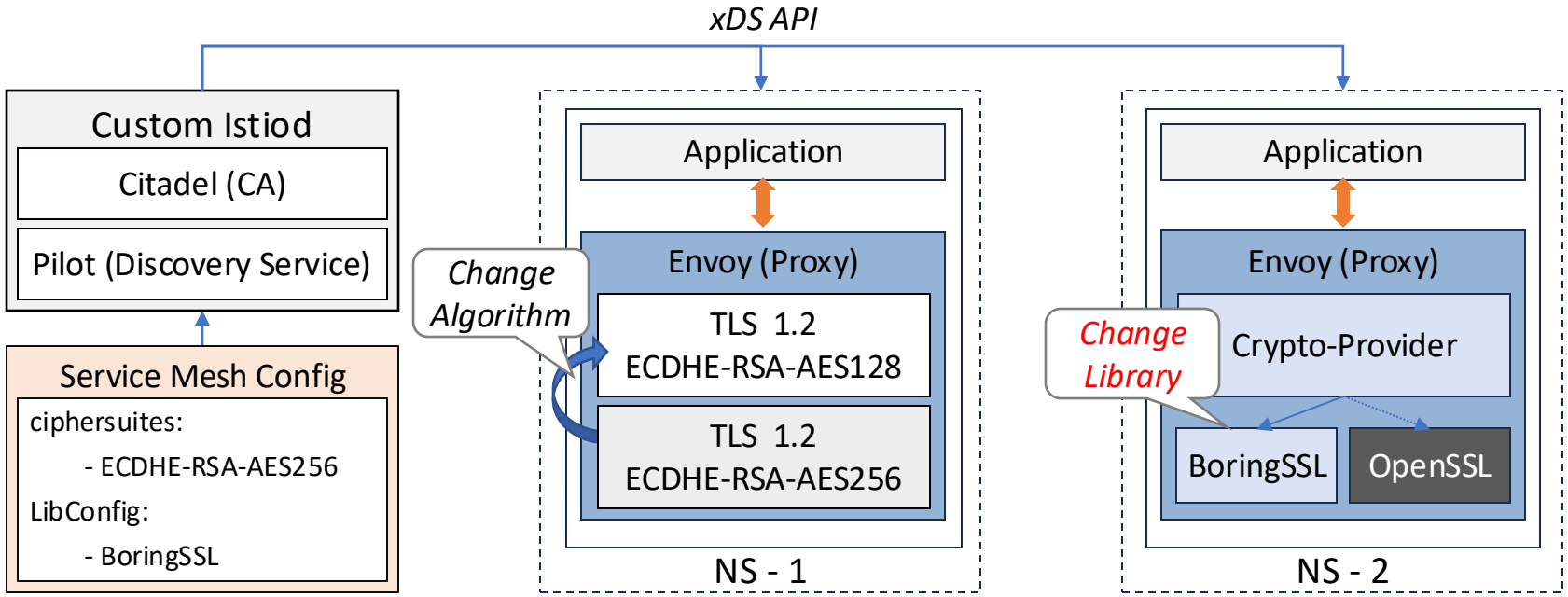
Namespace-level:
ECDHE-RSA-AES128-GCM-SHA256

Label-level:
ECDHE-ECDSA-AES128-GCM-SHA256

암호화 민첩성을 위한 암호 전환 방안 #2

- **암호화 라이브러리 동적 교체**
  - 새로운 암호화 알고리즘 적용과 취약점 대응을 위한 암호화 모듈(라이브러리) 교체 용이성 확보 필요
    - 이를 통해 BoringSSL, OpenSSL과 같은 특정 암호화 라이브러리에 대한 종속성 최소화
  - **Crypto Provider**를 통한 라이브러리 추상화 계층 활용

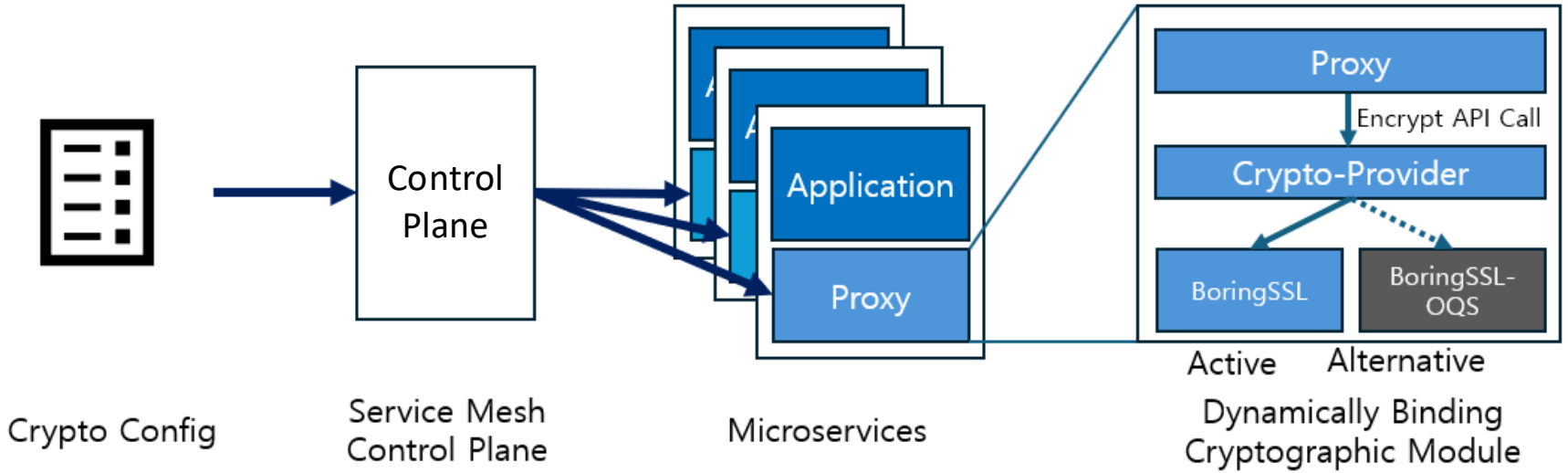
  - **특정 암호화 라이브러리에 대한 종속성 최소화 및 다양한 암호화 라이브러리 간 신속한 전환 가능**



암호화 민첩성을 위한 암호 전환 방안 #3

- **명시적 암호화 정책 설정**
  - 암호화 정책 설정을 통해 암호화 구성요소를 명확하게 정의
    - 암호화 알고리즘, 키 길이, 프로토콜 버전 등을 지정
  - 이를 참조하는 **Crypto-Provider**를 활용하여 암호화 라이브러리의 동적 전환을 실현
  - 코드 수정 없이 새로운 암호화 라이브러리로 교체 가능

  - **암호화 알고리즘 수명 주기 관리 및 취약점 분석을 용이하게 하며, 일관된 보안 정책을 적용할 수 있음**



맺음말

- **클라우드 환경에서의 암호화 체계 가시화**
  - 인증서 및 키 관리 전 과정의 실시간 모니터링 및 가시화
  - 마이크로서비스 간 암호화 통신 탐지 및 취약성 분석
  - 클라우드 환경 전반에 걸친 암호화 체계 가시성 확보

- **클라우드 환경에서의 암호 전환 방안**
  - 명시적 암호화 정책 설정
  - 암호화 알고리즘 동적 적용
  - 암호화 라이브러리 동적 교체

감사합니다

남 재 현 (namjh@dankook.ac.kr)