저장 데이터 암호화

Ken Beer

Ryan

Holland

2014년 11월

목차

요약

조직의 정책 또는 업계나 정부 규범에 따라 저장된 암호화를 사용하여 데이터를 보호해야 할 수 있습니다. Amazon Web Services(AWS)의 탄력적인 특성이 사용자의 요구 사항에 맞는 다양한 옵션을 제공합니다. 본 백서에서는 오늘날 제공되는 다양한 저장 데이터 암호화 방법을 간략히 살펴봅니다.

서론

Amazon Web Services(AWS)는 높은 가용성을 갖춘 안전하고 확장 가능한 클라우드 컴퓨팅 플랫폼을 제공하며, 이를 통해 고객들이 다양한 애플리케이션을 구축할 수 있도록 유연성을 제공합니다. 클라우드에 저장하는 데이터를 한 층 더 보호하고 싶은 경우, 저장된 데이터를 암호화하는 여러 가지 옵션을 고려해 볼 수 있습니다. 이러한 옵션은 완전히 자동화된 AWS 암호화 솔루션부터 수동의 클라이언트 측 옵션까지 다양합니다. 올바른 솔루션을 선택하는 것은 사용자가 사용하고 있는 AWS 서비스 및 키 관리 요구 사항에 따라 다릅니다. 본 백서에서는 AWS에서의 다양한 저장 데이터 암호화 방법을 간략히 살펴봅니다. 기술된 암호화 기법을 어떻게 구현하는지에 대해 더 자세히 알고 싶으시면 추가 리소스의 링크를 참조하십시오.

암호화의 키: 누가 키를 제어하는가?

어떤 시스템에서든 암호화라고 하면 다음 세 가지 요소를 필요로합니다. (1) 암호화할 데이터, (2) 데이터를 암호화하는 방법 (암호화 알고리즘), (3) 데이터 및 알고리즘과 함께 사용될 암호화 키가 그것입니다. 대부분의 최신 프로그래밍 언어는 Advanced Encryption Standard(AES)를 포함한 다양한 종류의 암호화 알고리즘을 활용할 수 있는 라이브러리를 제공합니다. 적절한 알고리즘을 선택하기 위해서는 애플리케이션별 요구 사항에 부합하는 보안 특성, 성능 및 규정 준수 (Compliance) 사항을 평가해 보아야 합니다. 암호화 알고리즘을 선택하는 것과 더불어, 허가되지 않은 접근으로부터 키를 보호하는 것이 매우 중요합니다. 암호화 키의 보안 관리는 종종 키 관리 인프라(KMI)를 통해 수행됩니다. KMI는 일반 텍스트 키를 보호하는 스토리지 계층 및 키 사용을 허가하는 관리 계층 등 2개의 하위 구성 요소로 구성됩니다. KMI에서 키를 보호하는 일반적인 방법은 하드웨어 보안 모듈(HSM)을 사용하는 것입니다. HSM은 디바이스의 키를 사용하여 암호화 작업을 수행하는 전용 스토리지 및 데이터 처리 디바이스입니다. HSM은 일반적으로 무단 사용으로부터 키를 보호하기 위해 증거 훼손 방지 또는 저항을 제공합니다. 소프트웨어 기반 권한 부여 계층은 HSM을 관리할 수 있는 사용자 및 어떤 사용자 또는 애플리케이션이 HSM의 어떤 키를 사용할 수 있는지를 제어합니다.

AWS 내의 다양한 데이터 계층에 암호화를 적용할 때에는 누가 어떤 조건에서 암호화 키나 데이터에 접근 가능한지 정확하게 파악하는 것이 중요합니다. 그림 1에서와 같이 사용자 및/또는 AWS가 암호화 방법 및 KMI를 제공하는 방법에는 3가지 모델이 있습니다.

- 사용자가 암호화 방법 및 전체 KMI를 제어합니다.
- 사용자가 암호화 방법을 제어하고, AWS가 KMI 스토리지 구성 요소를 제공하고, 사용자가 KMI 관리 계층을 제공합니다.
- AWS가 암호화 방법 및 전체 KMI를 제어합니다.

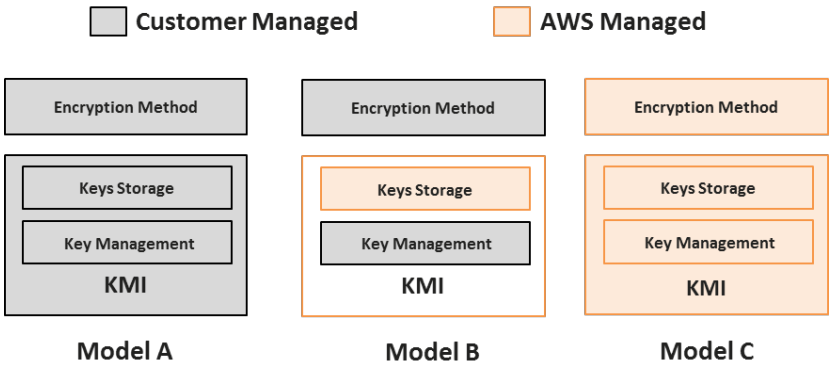


그림 1: AWS의 암호화 모델

모델 A: 사용자가 암호화 방법 및 전체 KMI 제어

이 모델에서는 사용자가 애플리케이션의 암호화 방법을 모두 통제할 뿐만 아니라, 사용자 자신의 KMI를 사용하여 키를 생성, 저장하고 접근을 관리합니다. KMI 및 암호화 방법의 물리적 위치는 AWS 외부 또는 사용자가 소유한 Amazon Elastic Compute Cloud(EC2) 인스턴스 내부가 될 수 있습니다. 암호화 방법은 오픈 소스 도구, AWS SDK 또는 타사 소프트웨어 및/또는 하드웨어의 조합이 될 수 있습니다. 이 모델의 중요한 보안 속성은 사용자가 암호화 키 및 암호화 코드에서 이러한 키를 활용하는 실행 환경에 대한 전권을 가진다는 것입니다. AWS는 키에 액세스할 수 없으며 사용자를 대신하여 암호화나 복호화를 수행할 수 없습니다. 사용자가 키의 올바른 저장, 관리 및 사용을 담당하고 데이터의 기밀성, 무결성, 가용성을 보장해야 합니다. 다음 섹션에서 설명하는 대로 AWS에서 데이터를 암호화할 수 있습니다.

Amazon S3

사용자가 원하는 모든 암호화 방법을 사용하여 데이터를 암호화한 다음, Amazon Simple Storage Service(S3) API를 사용하여 암호화된 데이터를 업로드할 수 있습니다. 대부분의 일반 애플리케이션 언어에는 사용자가 애플리케이션에서 암호화를 수행할 수 있게 해주는 암호화 라이브러리가 들어 있습니다. 일반적으로 많이 사용하는 오픈 소스 도구로는 Bouncy Castle 및 OpenSSL 등 2가지가 있습니다. 객체를 암호화한 다음, KMI에 키를 안전하게 저장하면 암호화된 객체가 PUT 요청과 함께 바로 Amazon S3로 업로드됩니다. 이 데이터를 복호화하려면 Amazon S3 API에서 GET 요청을 수행한 다음, 암호화 데이터를 복호화할 로컬 애플리케이션으로 전달합니다.

AWS는 Amazon S3 암호화 클라이언트와 함께 이러한 오픈 소스 암호화 도구의 대안을 제공하는데, 이는 AWS SDK에 포함되어 있는 오픈 소스 API 세트입니다. 이 클라이언트를 활용함으로써, Amazon S3 호출 과정 중 필요한 암/복호화 키를 사용자의 KMI로부터 제공할 수 있습니다SDK는 애플리케이션 내에서 Java Cryptographiy Extensions(JCEs)을 활용하며, 이는 대칭키 혹은 비대칭키를 입력받아 객체를 암호화한 다음 Amazon s3로 업로드합니다. SDK를 객체를 검색하는 데 사용하는 경우에는 절차가 반대입니다. Amazon S3로부터 다운로드된 암호화 객체는 KMI의 키와 함께 클라이언트로 전달됩니다. 애플리케이션의 기본 JCE는 객체를 복호화합니다.

Amazon S3 암호화 클라이언트는 Java, Ruby 및 .NET 용 AWS SDK로 통합되어, Amazon S3와 통신하기 위하여 작성된 사용자 애플리케이션 내의 예전 암호화 코드를 수정없이 투명하게 바꿔칠 수 있습니다. AWS에서 암호화 방법을 제공하더라도 사용자가 사용할 엔진에 대해 키를 제어하므로 사용자가 데이터 보안을 제어합니다. 사용자가 Amazon S3 암호화 클라이언트를 On-Premise에서 사용할 경우, AWS는 사용자의 키나 암호화되지 않은 데이터에는 접근할 수 없습니다. Amazon EC2에서 실행되는 애플리케이션에서 클라이언트를 사용하는 경우, 모범 사례는 KMI로부터 보안 전송(예: Secure Sockets Layer(SSL) 또는 Secure Shell(SSH))을 사용하여 클라이언트로 키를 전달하여 기밀성을 보장하는 것입니다. 자세한 내용은 Java용 AWS SDK 설명서 및 클라이언트 측 암호화 사용(*Amazon S3 개발자 설명서*)을 참조하십시오. 그림 2에서는 이러한 2가지 클라이언트 측 암호화 방법이 Amazon S3 데이터를 보호하는 방법에 대해 보여줍니다.

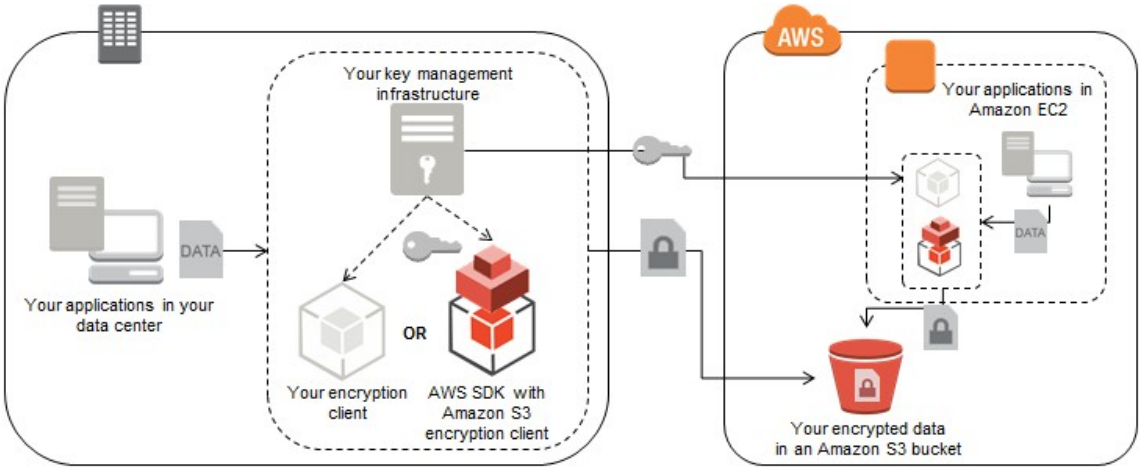


그림 2: 온프레미스 시스템 또는 Amazon EC2 애플리케이션 내에서 Amazon S3 클라이언트 측 암호화

Amazon S3로 이동할 데이터를 암호화할 때 키 관리 프로세스를 간소화할 수 있는 타사 솔루션을 사용할 수 있습니다. CloudBerry Explorer PRO for Amazon S3 및 CloudBerry Backup 두 솔루션 모두 사용자가 입력한 암호를 암호화 체계에 적용하여 Amazon S3에 저장된 파일을 보호하는 클라이언트 측 암호화 옵션을 제공합니다. 프로그래밍적 암호화가 필요한 경우 Java용 SafeNet ProtectApp이 SafeNet KeySecure KMI와 통합되어 애플리케이션에서 클라이언트 측 암호화를 제공합니다. KeySecure KMI는 안전한 키 스토리지 및 AWS SDK와 호환되는 ProtectApp Java 클라이언트로 전달되는 키에 대한 정책 설정을 제공합니다. KeySecure KMI는 온프레미스 어플라이언스 또는 Amazon EC2의 가상 어플라이언스로 실행될 수 있습니다. 그림 3에서는 SafeNet 솔루션을 사용하여 Amazon S3에 저장된 데이터를 암호화하는 방법을 보여줍니다.

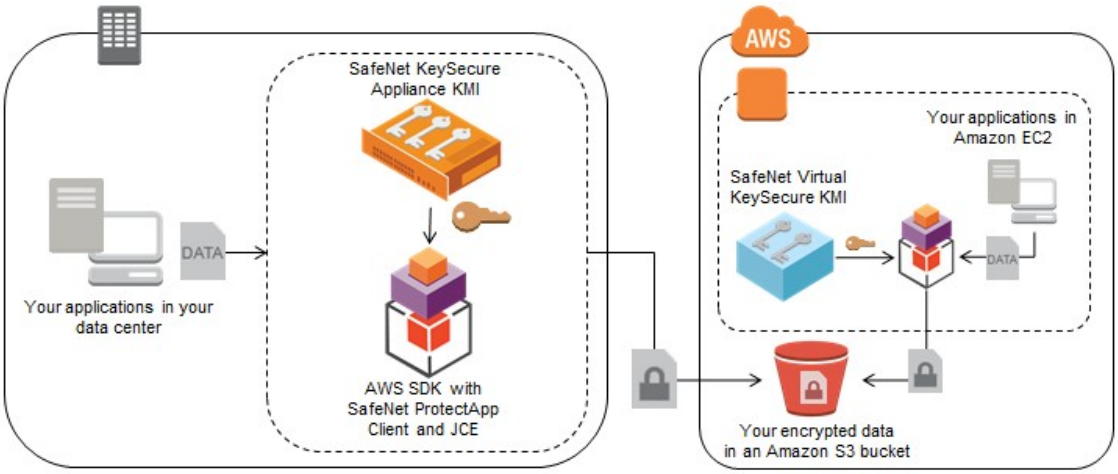


그림 3: SafeNet ProtectApp 및 SafeNet KeySecure KMI를 사용하여 온프레미스 시스템 또는 Amazon EC2의 애플리케이션 내에서 Amazon S3 클라이언트 측 암호화

Amazon EBS

Amazon Elastic Block Store(EBS)는 Amazon EC2 인스턴스와 함께 사용하기 위해 블록 수준 스토리지 볼륨을 제공합니다. Amazon EBS 볼륨은 네트워크에 연결되어 있으며, 인스턴스 수명과 관계없이 지속됩니다.

Amazon EBS 볼륨은 인스턴스에 블록 디바이스로 제공되므로 파일 시스템 수준 또는 블록 수준 암호화를 위해 대부분의 표준 암호화 도구를 활용할 수 있습니다. Linux용으로 제공되는 일반 블록 수준 오픈 소스 암호화 솔루션으로는 *Loop-AES*, *dm-crypt*(*LUKS* 포함 또는 미포함) 및 *TrueCrypt*가 있습니다. 이 솔루션은 각각 커널 스페이스 디바이스 드라이버를 사용하여 파일 시스템 계층 아래에서 작동하여 데이터의 암호화 및 복호화를 수행합니다. 데이터가 저장되는 디렉토리에 상관없이 볼륨에 기록되는 모든 데이터를 암호화하고자 할 경우 이러한 도구가 유용합니다.

또 다른 방법으로는 파일 시스템 수준 암호화 사용을 들 수 있는데, 이 솔루션은 기존 파일 시스템 위에 암호화된 파일 시스템을 올려 쌓는 방법입니다. 이 방법은 특정 디렉터리를 암호화할 때 일반적으로 사용됩니다. 파일 시스템 수준 암호화 도구의 Linux 기반 오픈 소스 예로는 *eCryptfs* 및 *EncFs* 의 두 가지를 들 수 있습니다.

이 솔루션이 작동하려면 사용자가 수동으로 또는 KMI로부터 키를 제공해야 합니다. 블록 수준 및 파일 시스템 수준 암호화 도구를 사용할 때 주의해야 할 중요한 점 중 하나는 Amazon EBS 부트 볼륨이 아닌 데이터 볼륨만 암호화하는 데 사용해야 한다는 점입니다. 이러한 도구가 시스템 기동 시 부트 볼륨에 사용할 수 있는 신뢰된 키를 자동으로 생성할 수 있는 기능을 지원하지 않기 때문입니다.

Windows 인스턴스에 연결된 Amazon EBS 볼륨을 암호화하는 작업은 TrueCrypt 같은 오픈 소스 애플리케이션뿐만 아니라 *BitLocker* 또는 *Encrypted File System(EFS)*을 사용해서도 수행할 수 있습니다. 어느 경우든 여전히 사용자가 이러한 암호화 방법에 키를 제공해야 하며 데이터 볼륨만 암호화할 수 있습니다.

AWS 파트너 솔루션 중에는 필요한 암호화 키를 제공하고 이를 보호하며, Amazon EBS 볼륨의 암호화 과정을 자동화 해주는 솔루션도 있습니다. Trend Micro SecureCloud 및 SafeNet ProtectV는 Amazon EBS 볼륨을 암호화하고 KMI 기능도 함께 제공하는 파트너 제품 중 두 가지입니다. 두 제품 모두 데이터 볼륨뿐 아니라 부트 볼륨도 암호화할 수 있습니다. 이 솔루션들은 Amazon EBS 볼륨이 Auto Scaling 그룹에 포함된 EC2 인스턴스에 부착된(Attach) 경우도 지원합니다. 그림 4는 SaaS(Software as a Service) 형태로 온프레미스에서 관리되거나, Amazon EC2에서 실행되는 소프트웨어 내의 키를 사용하여 SafeNet 및 Trend Micro 솔루션이 Amazon EBS에 저장된 데이터를 암호화하는 방법을 보여줍니다.

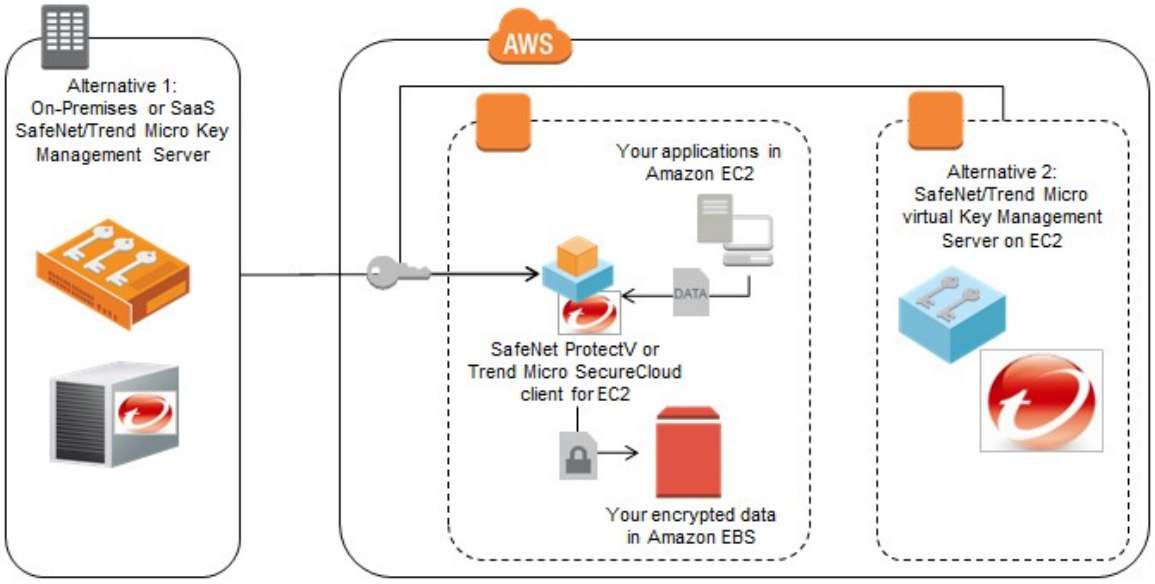


그림 4: SafeNet ProtectV 또는 Trend Micro SecureCloud를 사용한 Amazon EBS의 암호화

AWS Storage Gateway

AWS Storage Gateway는 온프레미스 소프트웨어 어플라이언스를 Amazon S3와 연결하는 서비스입니다. 이것(Storage Gateway)은 네트워크 상에서 iSCSI 디스크로 표시됨으로써 다른 소스로부터 데이터를 편리하게 복사할 수 있습니다. AWS Storage Gateway에 연결된 디스크 볼륨의 데이터는 정책에 따라 Amazon S3로 자동 업로드 됩니다. 소스 데이터가 디스크로 이동하기 전에 이전에 설명한 파일 암호화 방법(예: Bouncy Castle 또는 OpenSSL) 중 하나를 사용하여 디스크 볼륨의 소스 데이터를 암호화할 수 있습니다. 또는 AWS Storage Gateway가 노출하는 iSCSI 엔드포인트 상에서 블록 수준 암호화 도구(예: BitLocker나 dm-crypt/LUKS)를 사용하여 디스크 볼륨의 모든 데이터를 암호화할 수 있습니다. 그 밖에도 AWS 파트너 솔루션인, Trend Micro SecureCloud 및 SafeNet StorageSecure도 AWS Storage Gateway에서 노출하는 iSCSI 디스크 볼륨에 대해 암호화 및 키 관리를 수행할 수 있습니다. 이들 파트너는 데이터를 암호화하고 필요한 키를 관리하기 용이한 솔루션을 제공하는데, Amazon EBS 암호화 솔루션이 동작하는 방식과 설계면에서 유사합니다.

Amazon RDS

클라이언트 측 기술을 사용하여 Amazon Relational Database Service(RDS)에서 데이터를 암호화하려면 원하는 데이터 쿼리 작동 방식을 고려해야 합니다. Amazon RDS는 데이터 저장에 사용되는 연결된 디스크를 노출하지 않으므로, 앞서 Amazon EBS 섹션에서 기술한 투명한 디스크 암호화를 사용할 수 없습니다. 그러나 데이터가 Amazon RDS 인스턴스에 전달되기 전에 이전에 설명한 표준 암호화 라이브러리(예: Bouncy Castle, OpenSSL)를 사용하여 애플리케이션의 데이터베이스 필드를 선택적으로 암호화하는 것은 가능합니다. 이 특정 필드 데이터가 암호화 됨에 따라 범위 검색 쿼리를 사용하기는 용이하지 않지만, 암호화되지 않은 필드를 기반으로 하는 쿼리는 여전히 유용한 결과를 반환합니다. 로컬 애플리케이션을 사용하여 반환된 결과의 암호화된 필드를 복호화하여 표시할 수 있습니다. 암호화된 데이터에 대해 보다 효율적으로 쿼리를 수행하려면, 키가 추가된(Keyed) HMAC(Hash-based Message Authentication Code) 값을 스키마에 저장하고 이를 위해 해시 함수에 키를 전달할 수 있습니다. 그리고 HMAC 필드에 인덱스를 생성할 수 있습니다. 이후 쿼리에서는 보호된 필드에 대해 HMAC 값으로만 대조가 됨으로써, 평문값을 쿼리에 노출시키지 않아도 됩니다. 이렇게 하면 쿼리의 일반 텍스트 값을 공개하지 않고도 데이터베이스에서 데이터베이스의 암호화된 데이터에 대해 쿼리를 수행할 수 있습니다. 어떤 암호화 방법을 선택하든 Amazon RDS 인스턴스로 전송되기 전에 사용자의 애플리케이션에서 수행되어야 합니다.

CipherCloud 및 Voltage Security는 Amazon RDS의 데이터 기밀성 보호 작업을 간소화해 주는 솔루션을 제공하는 두 AWS 파트너입니다. 두 공급업체 모두 스키마를 변경하지 않고 데이터베이스에 암호화 텍스트를 삽입할 수 있는 포맷유지암호화(FPE; Format-Preserving Encryption)를 사용한 데이터 암호화 기술을 보유하고 있습니다. 또한, 통합된 조회 테이블을 사용하는 토큰화 방식도 지원합니다. 어떤 경우든 데이터가 Amazon RDS 인스턴스에 기록되기 전에 애플리케이션에서 암호화 또는 토큰화됩니다. 이 두 파트너는 암호화 또는 토큰화된 필드가 있는 데이터베이스에 대해 인덱스를 설정하고 검색하는 기능을 제공합니다. 다른 애플리케이션에 키를 배포하거나 테이블을 매핑하여 암호화되거나 토큰화된 필드를 해제하지 않고도 해당 애플리케이션에서 암호화나 토큰화되지 않은 데이터를 데이터베이스로부터 읽을 수 있습니다. 예를 들어, Amazon RDS의 데이터를 Amazon Redshift 데이터 웨어하우징 솔루션으로 이동한 다음, 중요한 정보를 포함한 필드는 암호화나 토큰화가 유지된 상태에서 중요한 정보를 포함하지 않은 필드에 대해 쿼리를 수행할 수 있습니다. 그림 5는 Amazon EC2 내에서 Voltage 솔루션을 사용하여 Amazon RDS 인스턴스에 데이터가 기록되기 전에 해당 데이터를 암호화하는 방법을 보여줍니다. Amazon EC2의 애플리케이션에서 실행되는 Voltage Security 클라이언트가 데이터 센터에 있는 Voltage KMI로부터 암호화 키를 가져옵니다.

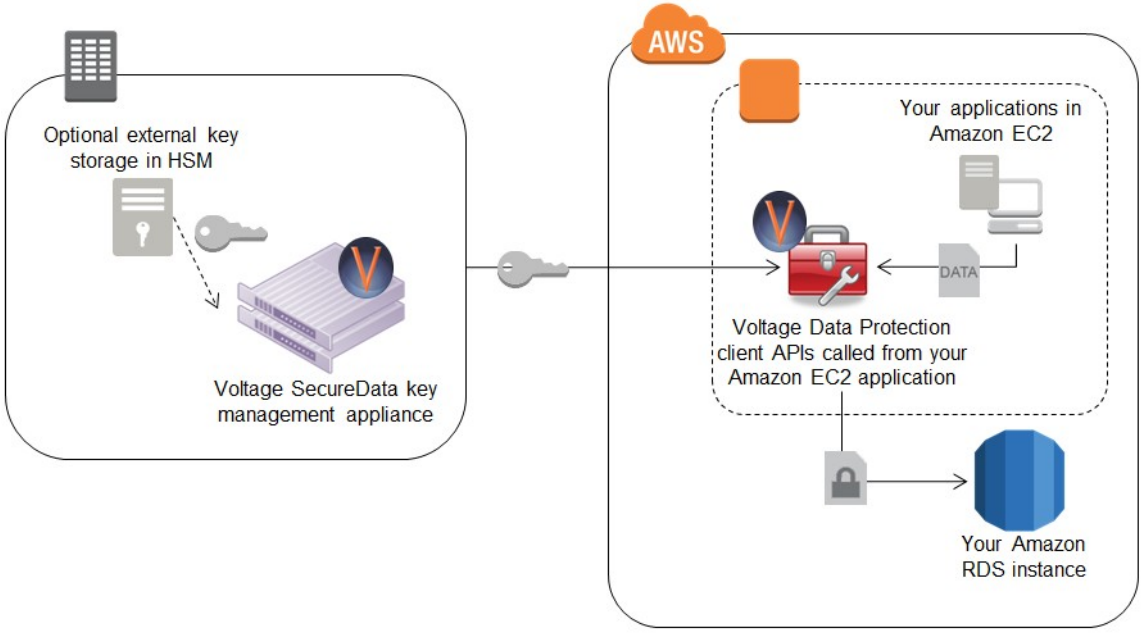


그림 5: Voltage SecureData를 사용하여 데이터를 Amazon RDS에 기록하기 전에 Amazon EC2 애플리케이션에 있는 해당 데이터 암호화

CipherCloud for Amazon Web Services는 Voltage Security 클라이언트가 Amazon EC2에서 실행되는 애플리케이션에 대해 작동하는 것과 유사한 방식으로 작동하는 솔루션으로, Amazon RDS로 암호화된 데이터를 보내고 받는 데 필요합니다. CipherCloud는 Amazon EC2에서 실행 중인지 아니면 데이터 센터에서 실행 중인지에 관계없이 애플리케이션에 설치될 수 있는 JDBC 드라이버를 제공합니다. 또한, CipherCloud for Any App 솔루션은 Amazon RDS 인스턴스로 보내거나 받는 중에 데이터를 가로채는 인라인 게이트웨이로서 배포될 수 있습니다. 그림 6은 이러한 방법으로 CipherCloud 솔루션을 배포하여, Amazon RDS 인스턴스로 데이터를 기록하기 전에, 데이터 센터 외부로 이동하는 데이터를 암호화하거나 토큰화하는 방법을 보여줍니다.

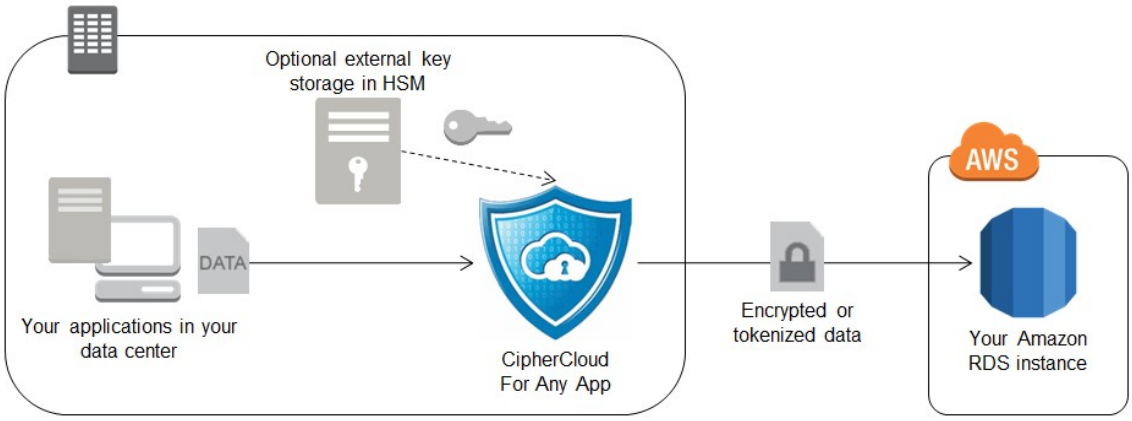


그림 6: CipherCloud 암호화 게이트웨이를 사용하여 데이터를 Amazon RDS에 기록하기 전에 데이터 센터에 있는 해당 데이터 암호화

Amazon EMR

Amazon Elastic MapReduce(EMR)는 Amazon EC2에서 실행되는 사용하기 쉬운 하둡 구현 방법을 제공합니다. MapReduce 작업을 통하여 암호화를 수행하는 과정에는 뚜렷하게 구분된 4가지의 관점에 따라 암호화 및 키 관리를 고려해야 한다.

1. 소스 데이터
2. 하둡 분산 파일 시스템(HDFS)
3. 셔플 단계
4. 출력 데이터

소스 데이터가 암호화되지 않은 경우 이 단계를 건너 뛸 수 있으며, Amazon EMR 클러스터로 전송되고 있는 데이터를 보호하기 위해 SSL이 사용될 수 있습니다. 소스 데이터가 암호화되어 있는 경우 데이터가 수집될 때 MapReduce 작업에서 해당 데이터를 복호화할 수 있어야 합니다. 작업 흐름에서 Java를 사용하고 소스 데이터가 Amazon S3에 있는 경우 앞의 Amazon S3 섹션에서 설명한 클라이언트 복호화 방법 중 하나를 사용할 수 있습니다.

HDFS 마운트 지점에 대해 사용된 스토리지는 클러스터 노드의 휘발성 스토리지입니다. 인스턴스 유형에 따라 두 개 이상의 마운트가 있을 수 있습니다. 이러한 마운트 지점을 암호화하려면 다음과 같은 작업을 하는 Amazon EMR 부트스트랩 스크립트를 사용해야 합니다.

- 하둡 서비스 중단
- 인스턴스에 파일 시스템 암호화 도구 설치
- 암호화된 디렉터리를 만들어 기존 마운트 지점 위에 암호화된 파일 시스템 마운트
- 하둡 서비스 재시작

예를 들어, 각 HDFS 마운트에 대해 오픈 소스 eCryptfs 패키지와 코드에서 생성된 휘발성 키를 사용하여 이러한 단계를 수행할 수 있습니다. 암호화되는 데이터가 HDFS 인스턴스의 수명 이상으로 지속되지 않으므로 이 암호화 키가 계속 저장되는 것을 걱정할 필요가 없습니다.

셔플 단계에는 감소 단계 전에 클러스터 노드 간 데이터가 전달되는 과정이 포함됩니다. 전송 중에 이러한 데이터를 암호화하려면 클러스터를 만들 때 하둡 부트스트랩 옵션을 구성하여 SSL을 설정하면 됩니다.

마지막으로, 출력 데이터를 암호화하려면 MapReduce 작업에서 KMI에서 나온 키를 사용하여 출력을 암호화해야 합니다. 암호화된 데이터를 Amazon S3로 전송하여 암호화된 형태로 저장할 수 있습니다.

모델 B: 사용자가 암호화 방법 제어, AWS가 KMI 스토리지 구성 요소 제공 및 사용자가 KMI 관리 계층 제공

이 모델은 사용자가 암호화 방법을 관리한다는 점에서 모델 A와 유사하지만, 키를 사용자가 온프레미스에서 관리하는 키 스토리지 시스템이 아닌 AWS CloudHSM 애플리케이션에 저장한다는 점이 모델 A와 다릅니다. AWS 환경에 키를 저장하지만, AWS의 어떤 직원도 이 키에 액세스할 수 없습니다. 사용자만 전용 HSM 내의 암호화 파티션에 액세스하여 키를 사용할 수 있기 때문입니다. AWS CloudHSM 어플라이언스는 탬퍼링(Tamper; 조작 혹은 훼손)을 감지하는 물리적/논리적 장치가 되어 있으며, 탬퍼링이 감지되면 장비(어플라이언스)를 초기화하도록 구성되어 있다. 초기화는 복호화 과정에 있는 키가 저장되는 HSM의 휘발성 메모리를 삭제하고 저장된 객체를 암호화하는 키를 손상시켜 실질적으로 HSM의 모든 키를 액세스할 수 없고 복구할 수 없는 상태로 만듭니다.

AWS CloudHSM 사용이 자신의 배포 상황에 맞는지 여부를 결정할 때는 데이터 암호화 과정에서 HSM의 역할을 이해하는 것이 중요합니다. HSM은 키 구성 요소를 생성하고 저장하는 데 사용할 수 있으며, 암호화 및 복호화 작업을 수행하지만, 키 수명 주기 관리 기능(예: 액세스 제어 정책, 키 로테이션)을 수행하지는 않습니다. 따라서 애플리케이션을 배포하기 전에 AWS CloudHSM 어플라이언스와 함께 호환 가능한 KMI도 필요할 수 있습니다. 사용자가 제공하는 KMI는 온프레미스에서 또는 Amazon EC2 내에서 배포될 수 있으며, SSL을 통해 AWS CloudHSM 인스턴스에 안전하게 통신하여 데이터 및 암호화 키를 보호할 수 있습니다. AWS CloudHSM 서비스가 SafeNet Luna 어플라이언스를 사용하므로, SafeNet Luna 플랫폼을 지원하는 키 관리 서버 또한 AWS CloudHSM과 함께 사용할 수 있습니다. 모델 A에서 AWS 서비스에 대해 설명한 암호화 방법은 솔루션에서 SafeNet Luna 플랫폼을 지원하는 한 모두 AWS CloudHSM과 함께 작동할 수 있습니다. 이에 따라 사용자는 트러스트 루트를 자신만 액세스할 수 있는 하드웨어 어플라이언스에 유지하면서 자신의 KMI를 AWS 컴퓨팅 환경에서 실행할 수 있습니다.

애플리케이션에서 Amazon Virtual Private Cloud(VPC)의 AWS CloudHSM 어플라이언스에 액세스할 수 있어야 합니다. SafeNet에서 제공하는 AWS CloudHSM 클라이언트는 AWS CloudHSM 어플라이언스와 상호 작용하여 애플리케이션의 데이터를 암호화합니다. 암호화된 데이터는 어느 Amazon S3로든지 전송하여 저장할 수 있습니다. 데이터베이스, 디스크 볼륨 및 파일 암호화 애플리케이션은 모두 AWS CloudHSM 및 사용자 지정 애플리케이션과 함께 지원될 수 있습니다. 그림 7에서는 AWS CloudHSM 솔루션이 Amazon VPC의 Amazon EC2에서 실행되는 애플리케이션과 함께 작동하는 방법을 보여줍니다.

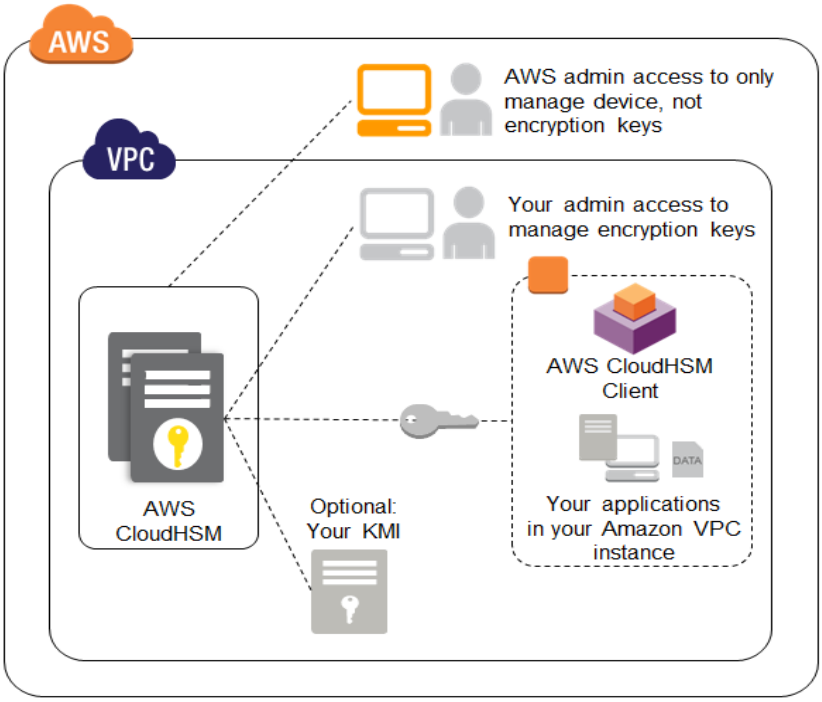


그림 7: Amazon VPC에 배포된 AWS CloudHSM

AWS CloudHSM 어플라이언스의 키에 대해 최고의 가용성 및 내구성을 원한다면 가용 영역 모두에 또는 사용자가 관리하는 온프레미스 SafeNet Luna 어플라이언스와 함께 여러 AWS CloudHSM 어플라이언스을 배포하는 것이 좋습니다. SafeNet Luna 솔루션은 어플라이언스 간에 키 구성 요소를 안전하게 복제하도록 해줍니다. 자세한 내용은 AWS 웹 사이트에서 AWS CloudHSM을 참조하십시오.

모델 C: AWS가 암호화 방법 및 전체 KMI 제어

이 모델에서는 AWS가 데이터의 서버 측 암호화를 제공하여 암호화 방법 및 키를 투명하게 관리합니다.

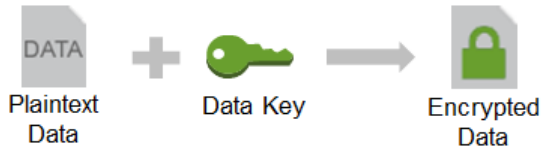
AWS Key Management Service(KMS)

AWS Key Management Service(KMS)는 관리형 암호화 서비스로, 키를 프로비저닝하고 사용하여 AWS 서비스에서 데이터 및 애플리케이션을 암호화하도록 해줍니다. HSM에서 마스터 키가 사용되는 방식과 유사하게 AWS KMS 내의 마스터 키가 사용됩니다. 마스터 키가 생성된 후에는 서비스에서 내보내지지 않도록 설계되었습니다. 데이터를 서비스로 전송한 후 사용자 계정의 특정 마스터 키를 사용하여 암호화 또는 복호화할 수 있습니다. 이러한 방식으로, 마스터 키에 접근하여 데이터를 암호화 및 복호화 할 수 있는 사람을 중앙집중적으로 통제할 수 있고, 이러한 접근을 감사할 수 있습니다. AWS KMS는 Amazon EBS, Amazon S3 및 Amazon Redshift를 포함한 다른 AWS 서비스와 함께 기본적으로 통합되어 있으므로 이러한 서비스 내의 데이터를 간단히 암호화할 수 있습니다. AWS SDK는 AWS KMS와 통합되어 사용자 지정 애플리케이션의 데이터를 암호화할 수 있습니다. 데이터를 암호화해야 하는 애플리케이션의 경우 AWS KMS가 전역적 가용성, 낮은 지연 시간 및 높은 수준의 내구성을 키에 제공합니다. 자세한 내용은 https://aws.amazon.com/kms/를 방문하거나 KMS 암호화 세부 정보 백서를 다운로드하십시오.

AWS KMS 및 데이터를 직접 암호화하는 기타 서비스는 봉투 암호화라는 방법을 사용하여 성능과 보안 간 균형을 유지합니다. 그림 8은 봉투 암호화를 설명합니다.

1. 데이터를 암호화하라는 요청을 하는 동시에 AWS 서비스에서 데이터 키를 생성합니다.

2. 데이터 키를 사용하여 데이터를 암호화합니다.



3. 데이터 키는 그 후 데이터를 저장하는 서비스에 고유한 키 암호화 키를 사용하여 암호화됩니다.

4. 이후 암호화된 데이터 키 및 암호화된 데이터가 AWS 스토리지 서비스에 저장됩니다.

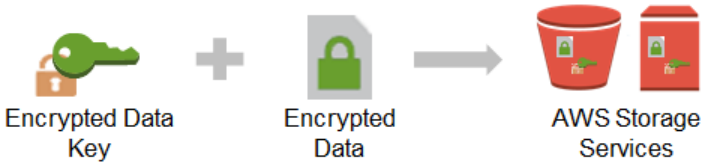


그림 8: 봉투 암호화

데이터 키를 암호화하는 데 사용된 키 암호화 키가 데이터 및 데이터 키와 별도로 저장되고 관리됩니다. 암호화 키에 대한 엄격한 액세스 제어가 이루어져 AWS 직원에 의한 무단 사용을 방지합니다. 평문 데이터가 필요할 경우에 이 프로세스를 반대로 수행하면 됩니다. 암호화된 데이터 키는 키 암호화 키를 사용하여 복호화되며, 데이터 키는 그 후 데이터를 복호화하는 데 사용됩니다.

다음 AWS 서비스에서 사용자가 선택할 수 있는 다양한 암호화 기능을 제공합니다.

Amazon S3

서버 측 암호화를 사용하여 Amazon S3의 데이터를 암호화하는 방법은 세 가지입니다.

1. **서버 측 암호화**: API 플래그를 설정하거나 AWS Management Console에서 확인란을 선택하여 Amazon S3의 디스크에 데이터가 기록되기 전에 해당 데이터를 암호화할 수 있습니다. 각 객체는 고유한 데이터 키로 암호화됩니다. 추가적인 보안을 위해 이 키는 Amazon S3에서 관리하는 주기적으로 로테이션되는 마스터 키로 암호화됩니다. Amazon S3 서버 측 암호화는 객체 및 마스터 키 모두에 256비트 Advanced Encryption Standard(AES) 키를 사용합니다. 이 기능을 사용하는 데는 Amazon S3 사용 요금 외에 추가 비용이 들지 않습니다.

2. **고객 제공 키를 사용하여 서버 측 암호화**: 객체를 Amazon S3로 업로드하는 동안 사용자 고유의 암호화 키를 사용할 수 있습니다. Amazon S3에서 이 암호화 키를 사용하여 AES-256을 통해 데이터를 암호화합니다. 객체가 암호화되면 사용자가 제공한 암호화 키는 데이터를 보호하기 위해 이 키를 사용한 Amazon S3 시스템에서 삭제됩니다. Amazon S3로부터 이 객체를 다시 읽어오기 위해서는 동일한 암호화 키를 제공해야 합니다. Amazon S3는 암호화 키가 일치하는지 확인한 후 객체를 복호화하고 객체를 사용자에게 반환합니다. 이 기능을 사용하는 데는 Amazon S3 사용 요금 외에 추가 비용이 들지 않습니다.

3. **KMS를 사용한 서버 측 암호화:** 객체를 최종적으로 암호화할 고유한 객체 키(그림 8에서 데이터 키로 지칭)를 암호화하는 데 사용하려는 계정 내에서 AWS KMS 마스터 키를 정의함으로써 Amazon S3의 데이터를 암호화할 수 있습니다. 객체를 업로드하면 KMS로 요청이 전송되어 객체 키가 만들어집니다. KMS는 이 객체 키를 생성하고 이를 앞서 지정한 마스터 키를 사용하여 암호화합니다. 그러면 KMS에서 이 암호화된 객체 키를 평문 객체 키와 함께 Amazon S3로 반환합니다. Amazon S3 웹 서버는 일반 텍스트 객체 키를 사용하여 객체를 암호화하고 방금 암호화한 객체를 저장(암호화된 객체 키와 함께)하고 메모리에서 일반 텍스트 객체 키를 삭제합니다. 이 암호화된 객체를 읽어오기 위해, Amazon S3는 AWS KMS에 암호화된 객체 키를 전송합니다. AWS KMS는 올바른 마스터 키를 사용하여 객체 키를 복호화하고 복호화된(일반 텍스트) 객체 키를 S3에 반환합니다. 평문 객체 키를 사용하여 S3는 암호화된 객체를 복호화하고 이를 사용자에게 반환합니다. 이 옵션의 사용 요금에 대한 자세한 내용은 AWS Key Management Service 요금 페이지를 참조하십시오.

Amazon EBS

Amazon EBS에서 볼륨을 생성할 때, AWS KMS 마스터 키를 사용하여 암호화하도록 선택할 수 있는데, 이는 EBS 볼륨을 최종적으로 암호화하는 고유 볼륨 키를 암호화하는데 쓰인다. 이렇게 선택하면 Amazon EC2 서버에서 인증된 요청을 AWS KMS로 전송하여 볼륨 키를 만듭니다. AWS KMS는 이 볼륨 키를 생성하고, 마스터 키를 사용하여 이를 암호화하고, 일반 텍스트 볼륨 키와 암호화된 볼륨 키를 Amazon EC2 서버로 반환합니다. 일반 텍스트 볼륨 키는 메모리에 저장되어, 연결된 EBS 볼륨으로 들어오거나 나가는 모든 데이터를 암호화 및 복호화합니다. 암호화된 볼륨(또는 해당 볼륨에서 만들어진 모든 암호화된 스냅샷)에서 인스턴스에 다시 연결해야 하는 경우 AWS KMS로의 호출이 만들어져 암호화된 볼륨 키를 복호화합니다. AWS KMS는 올바른 마스터 키를 사용하여 이 암호화된 볼륨 키를 복호화하고 복호화된 볼륨 키를 Amazon EC2에 반환합니다.

Amazon Glacier

데이터가 디스크에 기록되기 전에 Amazon Glacier 서비스 고유의 256 비트 AES키로 자동으로 암호화되며, 이 키는 AWS가 통제하는 별도 시스템에 저장됩니다. 이 기능을 사용하는 데는 Amazon Glacier 사용 요금 외에 추가 비용이 들지 않습니다.

AWS Storage Gateway

AWS Storage Gateway는 SSL을 통해 AWS에 데이터를 전송하고 Amazon S3 또는 Amazon Glacier에 저장된 암호화된 데이터를 각 서버 측 암호화 스키마를 사용하여 저장합니다.

Amazon EMR

*S3DistCp*는 대규모의 데이터를 Amazon S3에서 HDFS로, HDFS에서 Amazon S3로, Amazon S3 버킷 간에 이동하는 Amazon EMR 기능입니다. S3DistCp는 EMR 데이터를 사용자가 관리하는 Amazon S3 버킷에 기록할 때 서버 측 암호화를 사용하도록 Amazon S3에 요청하는 기능을 지원합니다. 이 기능을 사용하는 데는 Amazon S3를 사용하여 Amazon EMR 데이터를 저장하는 요금 외에 추가 비용이 들지 않습니다.

Amazon RDS의 Oracle

Amazon RDS의 Oracle에 대해 Oracle Advanced Security 옵션을 사용하도록 선택하여 내장 Transparent Data Encryption(TDE) 및 Native Network Encryption(NNE) 기능을 활용할 수 있습니다. Oracle 암호화 모듈은 데이터 및 키 암호화 키를 만들어 데이터베이스를 암호화합니다. Amazon RDS의 Oracle 인스턴스에 고유한 키 암호화 키는 주기적으로 로테이션되는 256비트 AES 마스터 키에 의해 자체적으로 암호화됩니다. 이 마스터 키는 Amazon RDS 서비스에 고유하며 AWS 제어를 받는 별도 시스템에 저장됩니다.

Amazon RDS의 Microsoft SQL Server

Amazon RDS의 Microsoft SQL Server에 Transparent Data Encryption(TDE)을 프로비저닝하도록 선택할 수 있습니다. SQL Server 암호화 모듈은 데이터 및 키 암호화 키를 만들어 데이터베이스를 암호화합니다. Amazon RDS의 SQL Server 인스턴스에 고유한 키 암호화 키는 주기적으로 로테이션되는 지역적 256비트 AES 마스터 키에 의해 자체적으로 암호화됩니다. 이 마스터 키는 Amazon RDS 서비스에 고유하며 AWS 제어를 받는 별도 시스템에 저장됩니다. 이 기능을 사용하는 데는 Amazon RDS의 Microsoft SQL Server 사용 요금 외에 추가 비용이 들지 않습니다.

Amazon Redshift

Amazon Redshift 클러스터를 만들 때 사용자 생성 테이블에서 모든 데이터를 암호화하도록 선택할 수 있습니다. Amazon Redshift 클러스터의 서버 측 암호화에서 선택할 수 있는 옵션은 3가지입니다.

1. 첫 번째 옵션에서는 데이터 블록(백업 포함)이 임의의 256비트 AES 키를 사용하여 암호화됩니다. 이러한 키는 임의의 256비트 AES 데이터베이스 키를 사용하여 자체적으로 암호화됩니다. 이 데이터베이스 키는 사용자의 클러스터에 고유한 256비트 AES 클러스터 마스터 키에 의해 암호화됩니다. 클러스터 마스터 키는 AWS 제어를 받는 별도 시스템에 저장된 Amazon Redshift 서비스에 고유한 주기적으로 로테이션되는 지역적 마스터 키로 암호화됩니다. 이 기능을 사용하는 데는 Amazon Redshift 사용 요금 외에 추가 비용이 들지 않습니다.

2. 두 번째 옵션을 사용하면 데이터베이스 키를 암호화하는 데 사용된 256비트 AES 클러스터 마스터 키가 AWS CloudHSM에서 또는 SafeNet Luna HSM 어플라이언스 온프레미스를 사용하여 생성됩니다. 이 클러스터 마스터 키는 그런 다음, HSM 외부로 이동하지 않는 마스터 키에 의해 암호화됩니다. Amazon Redshift 클러스터가 시작되면 클러스터 마스터 키가 HSM에서 복호화된 다음, 데이터베이스 키를 복호화하는 데 사용됩니다. 이 데이터베이스 키는 Amazon Redshift 호스트로 전송되어 클러스터 생명 주기 동안 메모리에만 상주합니다. 클러스터를 다시 시작하는 경우 클러스터 마스터 키가 다시 HSM으로부터 검색됩니다. 일반 텍스트 형태로 디스크에 저장되는 일은 없습니다. 이 옵션을 사용하면 데이터를 암호화하는 데 사용하는 키의 계층 구조 및 수명 주기를 보다 탄탄하게 제어할 수 있습니다. 이 기능을 사용하는 데는 Amazon Redshift(키 저장에 AWS CloudHSM을 사용하도록 선택하는 경우 해당 서비스 요금 추가) 사용 요금 외에 추가 비용이 들지 않습니다.

3. 세 번째 옵션에서는, 데이터베이스 키를 암호화하는 데 사용된 256비트 AES 클러스터 마스터 키가 AWS KMS에 생성됩니다. 이 클러스터 마스터 키는 그런 다음, AWS KMS 내의 마스터 키에 의해 암호화됩니다. Amazon Redshift 클러스터가 시작되면 클러스터 마스터 키가 AWS KMS에서 복호화된 다음, 데이터베이스 키를 복호화하는 데 사용됩니다. 이 데이터베이스 키는 Amazon Redshift 호스트로 전송되어 클러스터 생명 주기 동안 메모리에만 상주합니다. 클러스터를 다시 시작하는 경우 클러스터 마스터 키가 다시 AWS KMS의 강화된 보안 어플라이언스로부터 검색됩니다. 일반 텍스트 형태로 디스크에 저장되는 일은 없습니다. 이 옵션을 사용하면 마스터 키의 액세스 및 사용에 대해 세부적인 제어를 정의할 수 있고 AWS CloudTrail을 통해 이러한 제어를 감사할 수 있습니다. 이 옵션의 사용 요금에 대한 자세한 내용은 AWS Key Management Service 요금 페이지를 참조하십시오.

Amazon Redshift 클러스터 내에서 생성된 데이터 암호화 외에도, 사용자가 제공하는 키 및 Amazon S3 Encryption Client를 사용하여 이전에 암호화한 Amazon S3으로부터 Amazon Redshift로 암호화된 데이터를 로드할 수도 있습니다. Amazon Redshift는 Amazon S3 및 Amazon Redshift 간을 이동하는 데이터를 복호화하고 다시 암호화하여 데이터의 전체 수명 주기를 보호하는 기능을 제공합니다.

AWS의 여러 서비스 간 서버 측 암호화 기능을 사용하면 AWS Management Console에서 구성 설정을 하거나, 지정한 AWS 서비스에 대한 CLI나 API 요청을 만들어 간단히 데이터를 암호화할 수 있습니다. 암호화 키의 허가된 사용은 AWS에서 자동으로 안전하게 관리합니다. 이러한 키에 대한 허가되지 않은 접근은 데이터 유출로 이어질 수 있으므로, AWS는 허가되지 않은 접근을 최소화하는 강력한 접근 제어 기능을 갖춘 시스템 및 프로세스를 구축하였으며, 타사 감사 기관에 이러한 시스템의 검토를 요청하여 SOC 1, 2, 3, PCI-DSS 및 FedRAMP를 포함한 보안 인증을 취득하였습니다.

결론

암호화 키가 어떻게 관리되고 어디에 사용되는지에 대해 3가지 다른 모델을 알아보았습니다. 암호화 방법 및 KMI를 사용자가 모두 담당하는 경우, 애플리케이션의 데이터 암호화 방법에 대해 사용자가 세부적으로 제어할 수 있습니다. 그러나 이 방법은 배포 노력이 필요하고 애플리케이션의 암호화 방법과 AWS 서비스를 긴밀하게 통합할 수 없는 단점이 있습니다. 대안책으로, 배포가 쉽고 AWS 클라우드 서비스와 긴밀하게 통합되는 관리형 서비스를 선택할 수 있습니다. 이 옵션은 데이터를 저장하는 서비스를 위한 암호화, 사용자 키에 대한 통제, 키 저장소 보호, 그리고 모든 데이터 접근 시도에 대한 감사를 용이하게 한다.

표 1은 AWS 전반에 저장된 데이터를 암호화하는 데 사용할 수 있는 옵션을 요약해 보여줍니다. 사용하고 있는 AWS 서비스를 고려하여 자신의 데이터 분류에 가장 적합한 암호화 및 키 관리 모델이 어느 것인지 결정하는 것이 좋습니다.

	암호화 방법 및 KMI			
	모델 A		모델 B	모델 C
AWS 서비스	고객 관리형 키를 사용하는 클라이언트 측 솔루션	고객 관리형 키로 KMI를 사용하는 클라이언트 측 파트너 솔루션	AWS CloudHSM에서 고객 관리형 키를 사용하는 클라이언트 측 솔루션	AWS 관리형 키를 사용하는 서버 측 암호화
Amazon S3	Java용 AWS SDK의 Bouncy Castle, OpenSSL, Amazon S3 암호화 클라이언트	Java용 SafeNet ProtectApp	AWS CloudHSM 클라이언트와 통합된 사용자 지정 Amazon VPC-EC2 애플리케이션	Amazon S3 서버 측 암호화, 고객 제공 키를 사용하는 서버 측 암호화, AWS Key Management Service를 사용하는 서버 측 암호화
Amazon Glacier	해당 사항 없음	해당 사항 없음	AWS CloudHSM 클라이언트와 통합된 사용자 지정 Amazon VPC-EC2 애플리케이션	서버 측 암호화를 사용하여 모든 데이터가 자동으로 암호화됨
AWS Storage Gateway	Linux 블록 수준: - Loop-AES, dm-crypt(LUKS 포함 또는 미포함) 및 TrueCrypt Linux 파일 시스템: - eCryptfs 및 EncFs Windows 블록 수준: -TrueCrypt Windows 파일 시스템: - BitLocker	Trend Micro SecureCloud, SafeNet StorageSecure	해당 사항 없음	Amazon S3 서버 측 암호화

Amazon EBS	Linux 블록 수준: - Loop-AES, dm-crypt+LUKS 및 TrueCrypt Linux 파일 시스템: - eCryptfs 및 EncFs Windows 블록 수준: -TrueCrypt Windows 파일 시스템: - BitLocker, EFS	Trend Micro SecureCloud, SafeNet ProtectV	AWS CloudHSM 클라이언트와 통합된 사용자 지정 Amazon VPC-EC2 애플리케이션	AWS Key Management Service를 사용하는 Amazon EBS 암호화
Amazon RDS의 Oracle	Bouncy Castle, OpenSSL	CipherCloud Database Gateway 및 Voltage SecureData	AWS CloudHSM 클라이언트와 통합된 사용자 지정 Amazon VPC-EC2 애플리케이션	선택적 라이선스인 Oracle의 투명한 암호화 (Transparent Data Encryption; TDE) 및 네이티브 네트워크 암호화 (Native Network Encryption; NNE) Microsoft SQL Server의 TDE
Amazon RDS의 Microsoft SQL Server	Bouncy Castle, OpenSSL	CipherCloud Database Gateway 및 Voltage SecureData	AWS CloudHSM 클라이언트와 통합된 사용자 지정 Amazon VPC-EC2 애플리케이션	해당 사항 없음
Amazon Redshift	해당 사항 없음	해당 사항 없음	AWS CloudHSM 또는 온프레미스 Safenet Luna HSM에서 관리되는 마스터 키를 사용하는 암호화된 Amazon Redshift 클러스터	AWS 관리형 마스터 키를 사용하는 암호화된 Amazon Redshift 클러스터
Amazon EMR	eCryptfs		AWS CloudHSM 클라이언트와 통합된 사용자 지정 Amazon VPC-EC2 애플리케이션	저장된 데이터를 영구적으로 보호하기 위해 Amazon S3 서버 측 암호화를 사용하는 S3DistCp

표 1: 저장 데이터 암호화 옵션 요약

참조 자료

- Bouncy Castle Java 크립토 라이브러리
  http://www.bouncycastle.org/
- OpenSSL 크립토 라이브러리
  http://www.openssl.org/
- Amazon S3 암호화용 CloudBerry Explorer PRO
  http://www.cloudberrylab.com/amazon-s3-explorer-pro-cloudfront-IAM.aspx
- Java용 AWS SDK 및 Amazon S3를 사용하는 클라이언트 측 데이터 암호화
  http://aws.amazon.com/articles/2850096021478074

- Amazon S3, Amazon EBS 및 AWS CloudHSM용 SafeNet 암호화 제품
  http://www.safenet-inc.com/

- Trend Micro SecureCloud
  http://www.trendmicro.com/us/enterprise/cloud-solutions/secure-cloud/index.html

- CipherCloud for AWS 및 CipherCloud for Any App
  http://www.ciphercloud.com/

- Voltage Security SecureData Enterprise
  http://www.voltage.com/products/securedata-enterprise/

- AWS CloudHSM
  https://aws.amazon.com/cloudhsm/

- AWS Key Management Service
  https://aws.amazon.com/kms/

- Key Management Service 암호화 세부 정보 백서
  https://d0.awsstatic.com/whitepapers/KMS-Cryptographic-Details.pdf

- Amazon S3의 데이터를 암호화하기 위한 Amazon EMR S3DistCp
  http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/UsingEMR_s3distcp.html

- Amazon RDS의 Oracle용 투명한 데이터 암호화
  http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.Oracle.Options.html#Appendix.Oracle.Options.AdvSecurity

- Amazon RDS의 Microsoft SQL Server용 투명한 데이터 암호화
  http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html#SQLServer.Concepts.General.Options

- Amazon RedShift 암호화
  http://aws.amazon.com/redshift/faqs/#0210

- AWS 보안 블로그
  http://blogs.aws.amazon.com/security

문서 수정

2013년 11월: 초기 버전

2014년 11월:

- 모델 C의 AWS Key Management Service(KMS) 및 Amazon EBS에 섹션 추가
- 모델 C의 Amazon S3, Amazon Redshift 섹션 업데이트