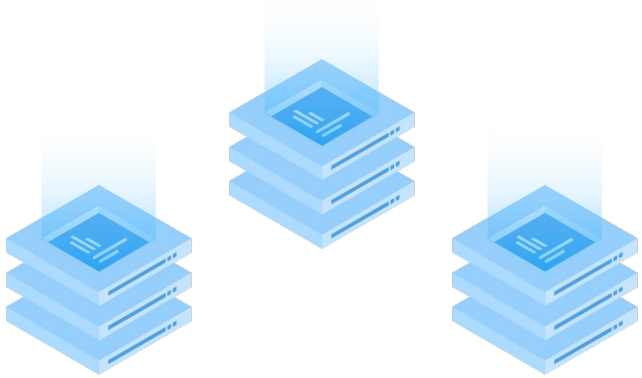
TiDB Security

Presented by Shuan Deng

PingCAP TiDB



Agenda

- 加密算法
- 传输层加密：TLS
- 静态加密：TDE
- 权限控制：RBAC
- 最佳实践

PingCAP

Part I - Introduction

场景

如何进入东升科技园 PingCAP 公司查看财务账单？

场景

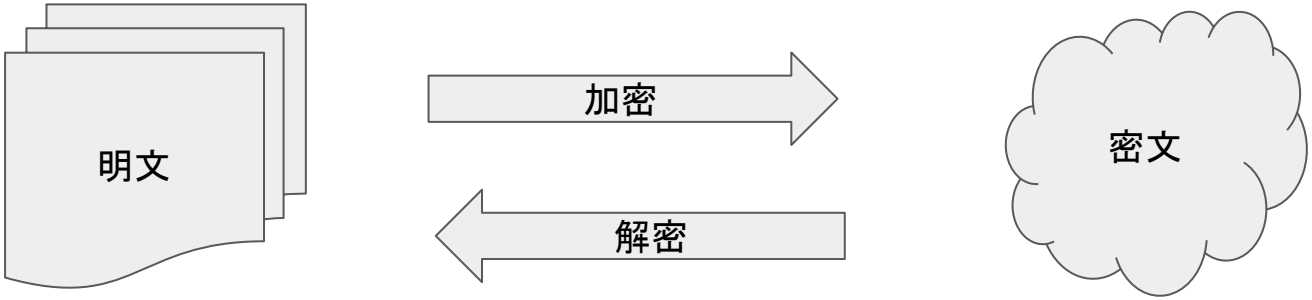
如何进入东升科技园 PingCAP 公司查看财务账单？

1. 进园区保安查园区卡
2. 进 PingCAP 公司刷门禁
3. 财务小姐姐确认身份
4. 财务小姐姐拿出钥匙开锁取出财务账单

PingCAP

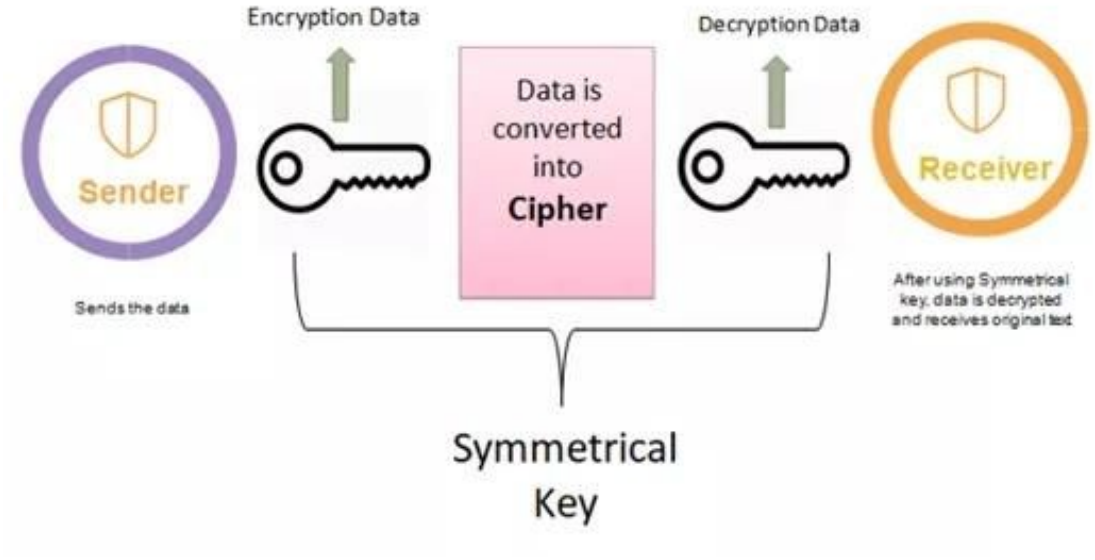
Part I - 加密算法

加密



对称加密

用单个密钥加密和解密数据，**性能好**。



利用对称加密，Server/Client 双方可以很好的加密和解密数据了。不知道密钥的中间用户即使拿到了数据也无法解密。

不过这里有个问题，Server 如何把这个密钥发送给 Client 呢？

明文传输密钥是不安全的。

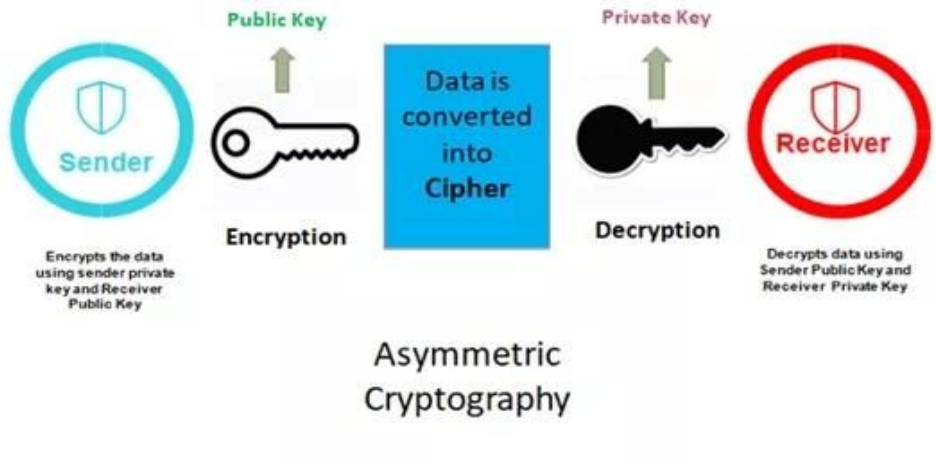
非对称加密

密钥对由公钥和私钥两部分组成，用公钥加密的数据只能由私钥解密，用私钥加密的数据只能由公钥解密，公钥是可以公开的，私钥要自己留好。**性能差**。

利用非对称加密，Server/Client 双方可以**协商**出一个对称加密的密钥，用来加密和解密数据。

非对称加密不适合用来加密和解密大量数据，因为性能差，而且也不安全。

因为公钥是公开的，拿到公钥的人都可以解密由私钥加密的数据。



PingCAP

Part II - 传输层加密（TLS）

网络传输问题

- Client 请求 Server 的时候，Server 会将自己的公钥发送给 Client。Client 端如何确定自己拿到的公钥确实是这个 Server 端的呢？怎么防止黑客在中间把公钥修改成自己的假公钥呢？

- Client 和 Server 如何利用非对称加密协商出一个对称加密的密钥呢？

数字证书

```
cjc:cn-pd changjunchang$ cat cert
-----BEGIN CERTIFICATE-----
MIIEKDCCAxCgAwIBAgIRAIHBg80UUxyJO33IinTKKrIwDQYJKoZIhvcNAQELBQAw
KTEVMBMGA1UEChMMY2VydC1tYW5hZ2VyMRAwDgYDVQQDEwdUaURCIENBMB4XDTIw
MDQwMTAzMzUxNVoXDTIwMDQwMTA0MzUxNVowHzEQMA4GA1UEChMHUGluZ0NBUDEL
MAkGA1UEAxMCUEQwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCtxWUM
OOe7gP6lt6LSPFE8tKbbo5aiK4cEsJiTwHWC5srh6mL8ph45bexyOqkwVC5JP36k
LknmX5fg/bPhdZj5c755HXvy5H9eml3NY6aacOU2LwDLCxvCv6I+7jFUR2OHZylC
nENnRPfhiioRQuExd8h+FOs4fsCW9/g61LMeEygO8VSZM9MaIMSO2IFCseKNgw6o
Djk6nBa7h26D4xUhNAM4x1+ez21UnBc6bsJb3MbdZfAOSW2PredgUV3/bF0XowLJ
CGk+G2OKTzzpHpeWYk9GY2CUkXnhLaWAiAhZWM8HyY3L4nkOCm1uZ3+yImPQfuZX
OtIDpjE8dvnyjY89AgMBAAGjggFTMIIBTzAOBgNVHQ8BAf8EBAMCBaAwDAYDVR0T
AQH/BAIwADCCAS0GA1UdEQSCASQwggEggg5jbHVzdGVyLXRscy1wZIIaY2x1c3Rl
ci10bHMtcGQuY2x1c3Rlci10bHOCHmNsdXN0ZXItdGxzLXBkLmNsdXN0ZXItdGxz
LnN2Y4ITY2x1c3Rlci10bHMtcGQtcGVlcoIfY2x1c3Rlci10bHMtcGQtcGVlci5j
bHVzdGVyLXRsc4IjY2x1c3Rlci10bHMtcGQtcGVlci5jbHVzdGVyLXRscy5zdmOC
FSouY2x1c3Rlci10bHMtcGQtcGVlcoIhKi5jbHVzdGVyLXRscy1wZC1wZWVyLmNs
dXN0ZXItdGxzgiUqLmNsdXN0ZXItdGxzLXBkLXBlZXIuY2x1c3Rlci10bHMuc3Zj
hwR/AAABhxAAAAAAAAAAAAAAAAAAAAABMA0GCSqGSIb3DQEBCwUAA4IBAQBtGY/k
CSOtnjxXbmsmag7iHLi0gGi535ojhWMNjXUPzf+xsuFmAq/aPJ8/vXSKBiWyIsIA
dUIeL8VILqITaIQQUX2Y/kWQNdrkZVrVSLYPVkKu2KJiStJsofYUKCjxfsFnFUUx
StTXdPZgr2IgmjXieF9WYKLbGuwCXL+qaAYinOdamuuZW36MNoRpPoqYH02/Jp3/
n4uw+NYTuVFztLaX3j+FKqod6cOWDQbnnf5fMxf70GmVPMGo7Zj/8fYNDme/mp+h
e4u0gh86H+4NpnRgDrCRg/BEQ6i44j+QIJ/2iZPbD9dnt4EuRrFQlB2zYYE24eKc
4+pYjX4hmVazHGZ8
-----END CERTIFICATE-----
```

其实 Client 请求 Server 的时候，Server 会将自己的**数字证书**发送给 Client。

数字证书通常来说是由受信任的数字证书颁发机构 CA（Certificate Authority），在验证服务器身份后颁发（签名），证书中包含公钥和所有者识别信息。数字证书被放到服务端，具有服务器身份验证和数据传输加密功能。

数字证书的主要目的就是提供公钥给用户用。用户可以根据 CA 来验证拿到的证书（内含公钥）确实是有效的。

数字证书

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            81:c1:83:cd:14:53:1c:89:3b:7d:c8:8a:74:ca:2a:b2
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: O=cert-manager, CN=TiDB CA
        Validity
            Not Before: Apr  1 03:35:15 2020 GMT
            Not After : Apr  1 04:35:15 2020 GMT
        Subject: O=PingCAP, CN=PD
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ad:c5:65:0c:38:e7:bb:80:fe:a5:b7:a2:d2:3c:
                    51:3c:b4:a6:db:a3:96:a2:2b:87:04:b0:98:93:c0:
                    75:82:e6:ca:e1:ea:62:fc:a6:1e:39:6d:ec:72:3a:
                    a9:30:54:2e:49:3f:7e:a4:2e:49:e6:5f:97:e0:fd:
                    b3:e1:75:98:f9:73:be:79:1d:7b:f2:e4:7f:5e:9a:
                    5d:cd:63:a6:9a:70:e5:36:2f:00:cb:0b:1b:c2:bf:
                    a2:3e:ee:31:54:47:63:87:67:29:42:9c:43:67:44:
                    f7:e1:8a:2a:11:42:e1:31:77:c8:7e:14:eb:38:7e:
                    c0:96:f7:f8:3a:d4:b3:1e:13:28:0e:f1:54:99:33:
                    d3:1a:20:c4:8e:d8:81:42:b1:e2:8d:83:0e:a8:0e:
                    39:3a:9c:16:bb:87:6e:83:e3:15:21:34:03:38:c7:
                    5f:9e:cf:6d:54:9c:17:3a:6e:c2:5b:dc:c6:dd:65:
                    f0:0e:49:6d:8f:ad:e7:60:51:5d:ff:6c:5d:17:a3:
                    02:c9:08:69:3e:1b:63:8a:4f:3c:e9:1e:97:96:62:
```

`openssl x509 -in tls.crt -noout -text`

有关用户**公钥**的信息，包括使用的算法和密钥本身的表示形式。

数字证书

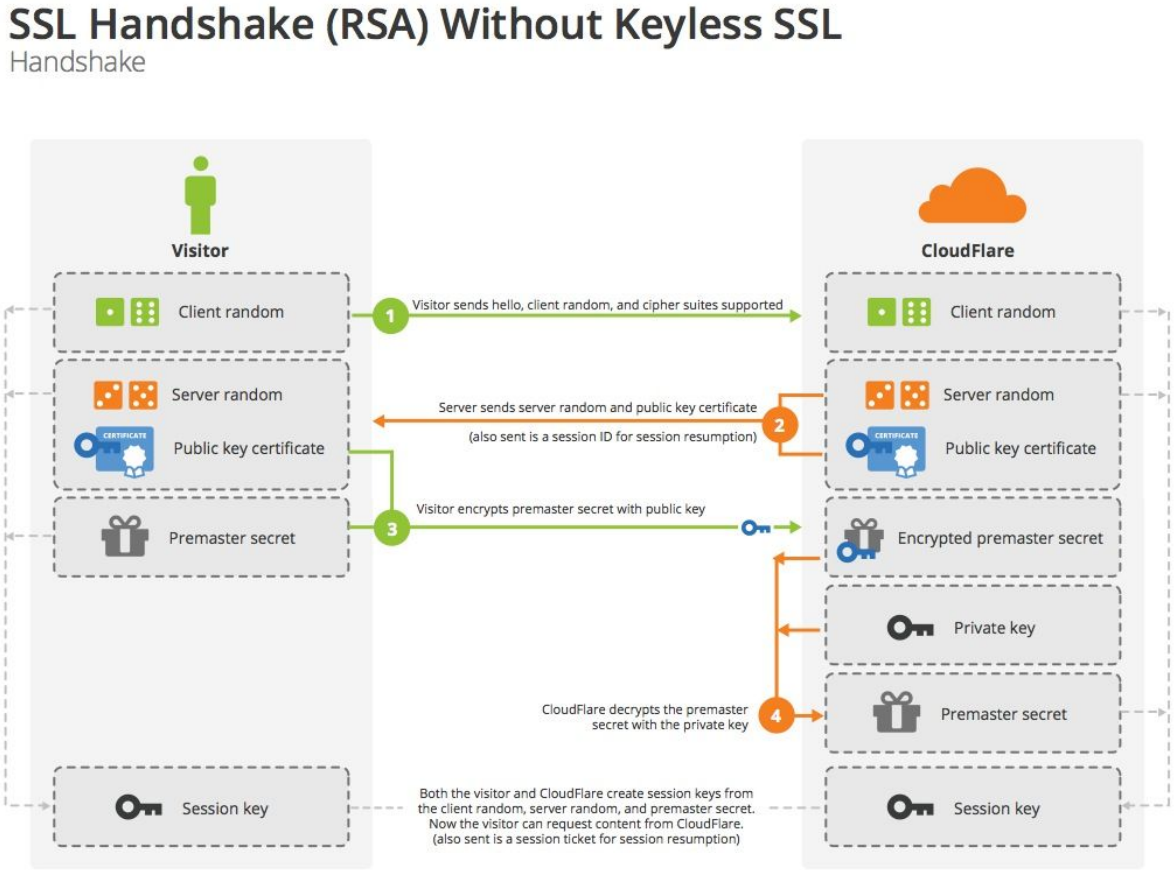
```
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Alternative Name:
                DNS:cluster-tls-pd, DNS:cluster-tls-pd.cluster-tls,
-pd-peer.cluster-tls.svc, DNS:*.cluster-tls-pd-peer, DNS:*.cluster-t
:1
    Signature Algorithm: sha256WithRSAEncryption
         6d:19:8f:e4:09:23:ad:9e:3c:57:6e:6b:26:6a:0e:e2:1c:b8:
         b4:80:68:b9:df:9a:23:85:63:0d:8d:75:0f:cd:ff:b1:b2:e1:
         66:02:af:da:3c:9f:3f:bd:74:8a:06:25:b2:22:c2:00:75:42:
         1e:2f:c5:48:2e:a2:13:68:84:10:51:7d:98:fe:45:90:35:da:
         e4:65:5a:d5:48:b6:0f:56:42:ae:d8:a2:62:4a:d2:6c:a1:f6:
         14:28:28:f1:7e:c1:67:15:45:31:4a:d4:d7:74:f6:60:af:62:
         20:9a:35:e2:78:5f:56:60:a2:db:1a:ec:02:5c:bf:aa:68:06:
         22:9c:e7:5a:9a:eb:99:5b:7e:8c:36:84:69:3e:8a:98:1f:4d:
         bf:26:9d:ff:9f:8b:b0:f8:d6:13:b9:51:73:b4:b6:97:de:3f:
         85:2a:aa:1d:e9:c3:96:0d:06:e7:9d:fe:5f:33:17:fb:d0:69:
         95:3c:c1:a8:ed:98:ff:f1:f6:0d:0e:67:bf:9a:9f:a1:7b:8b:
         b4:82:1f:3a:1f:ee:0d:a6:74:60:0e:b0:91:83:f0:44:43:a8:
         b8:e2:3f:90:20:9f:f6:89:93:db:0f:d7:67:b7:81:2e:46:b1:
         50:94:1d:b3:61:81:36:e1:e2:9c:e3:ea:58:8d:7e:21:99:56:
         b3:1c:66:7c
```

CA 的数字签名（Signature），是通过将证书中的所有数据散列在一起并用 CA 的**私钥**加密来获得的。

证书颁发机构 CA（Certificate Authority）

证书颁发机构（CA, Certificate Authority）即颁发数字证书的机构。是负责发放和管理数字证书的权威机构，并作为电子商务交易中受信任的第三方，承担公钥体系中公钥的合法性检验的责任。

TLS 握手（handshake）过程



PingCAP TiDB

TLS 握手（handshake）过程

握手的最终目的：双方协商出来一个**对称加密**密钥。

第一步，Client 请求 Server，附带一个随机数和客户端支持的加密方法给 Server。
第二步，Server 确认双方使用的加密方法，并给出**数字证书**、以及一个 Server 生成的随机数。
第三步，Client 确认数字证书有效，然后生成一个新的随机数，并使用数字证书中的公钥，加密这个随机数，发给 Server。
第四步，Server 使用自己的私钥，获取 Client 发来的随机数。
第五步，Client 和 Server 根据约定的加密方法，使用前面的三个随机数，生成"对话密钥"，用来加密接下来的整个对话过程。

Web TLS 证书

Server 向 CA 颁发机构申请证书：

a. Server 生成私钥
b. 用私钥签名 CSR
c. 将 CSR 提交给 CA
d. CA 用私钥给 server 签出公钥证书

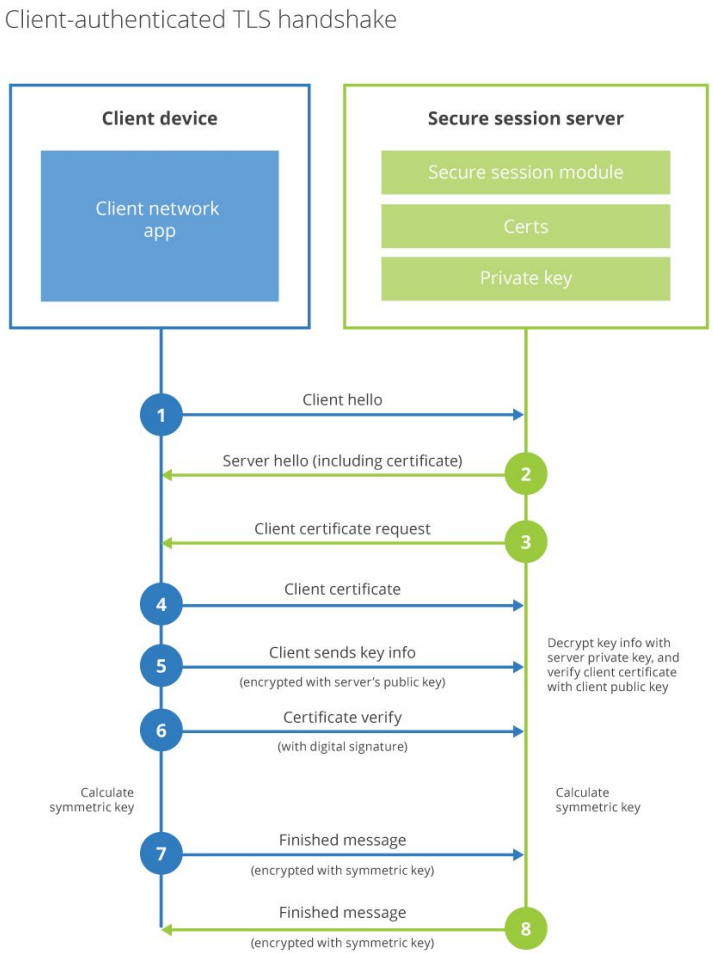
浏览器访问 Server：

a. Server 提供公钥
b. 浏览器对证书有效性进行验证

TiDB 集群双向认证（mTLS）

mTLS (mutual TLS)：双向 TLS 认证

通信双方都必须提供 TLS 证书



SQL 层 TLS

TiDB 集群 SQL 层 TLS 兼容 MySQL TLS 协议：

- Server 端不强制连接加密
- Server 端不强制客户端提供证书
- MySQL 5.7 客户端会优先选择加密连接
- MySQL 客户端默认不检查服务端证书的 CA，通过--sql-mode=VERIFY_IDENTITY/VERIFY_CA 强制验证
- TiDB 支持通过 TLS 进行认证

TLS 增强功能

- CommonName 校验
- 在线更新证书

PingCAP

Part III - 静态加密（TDE）

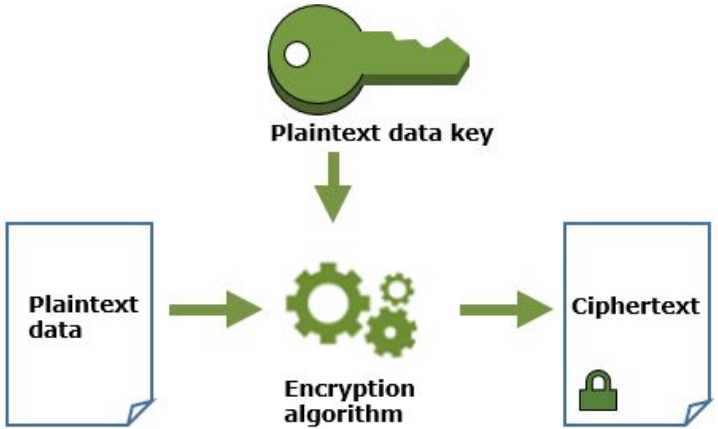
信封加密(envelope encryption)

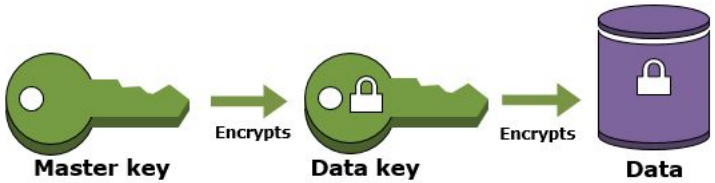
普通加密:

加密数据的密钥是明文的

信封加密:

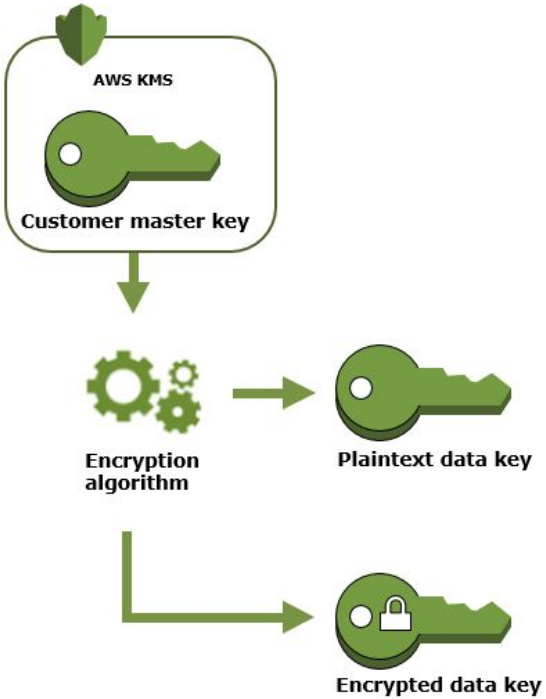
级联加密, 加密数据的密钥本身也是被加密的, 可以和数据保存在一起

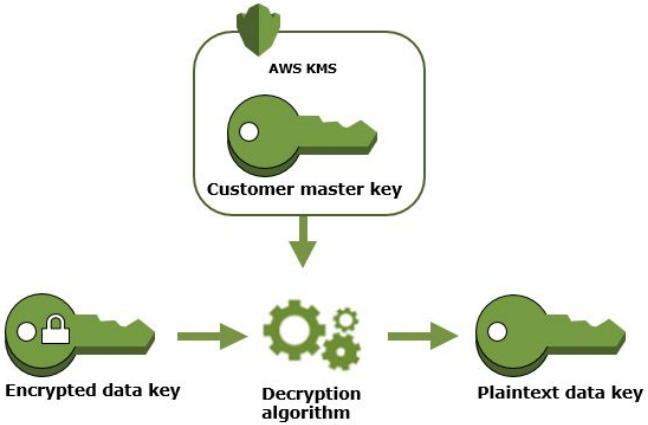




KMS

- GenerateDataKey
- Decrypt



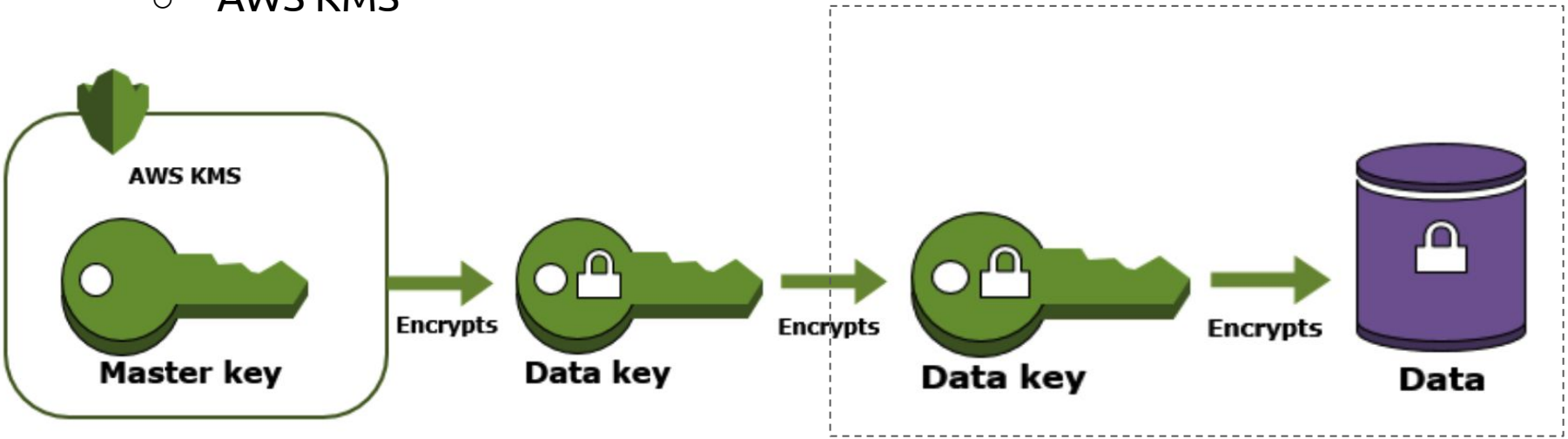


PingCAP TiDB

TiKV 静态加密

Transparent Data Encryption (Encryption at rest) RFC

- master key
  - Plaintext
  - File
  - AWS KMS



PingCAP

Part IV - 权限控制（RBAC）

权限控制

权限表：INFORMATION_SCHEMA.USER_PRIVILEGES

MySQL 5.7

1. create user
2. grant *privilege* to *user*

基于角色的权限控制（RBAC）

MySQL 8.0 RBAC

Role：一组权限的合集

1. create role
2. grant privilege to role
3. grant role to user

PingCAP

Part V - 最佳实践

安全最佳实践

- 最小权限原则(Least Privilege)
- 独立用户和证书
- 证书和密钥经常更新
- 开启 tidb-server TLS 认证替换密码认证

硬广

TiDB Operator 1.1 完善支持 TiDB 安全特性，并能够自动化签证书刷新证书

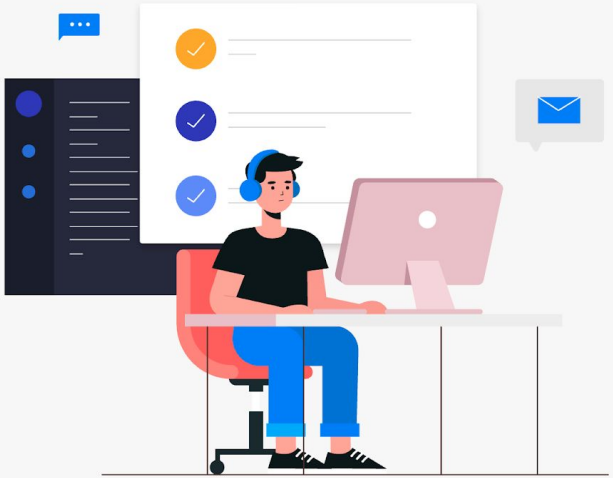
https://pingcap.com/docs-cn/tidb-in-kubernetes/stable/

https://github.com/pingcap/tidb-operator

欢迎使用

PingCAP TiDB

TiDB 4.0 捉“虫”竞赛



大赛流程

测试 Feature

我们为大家准备了用户文档，在开始测试前请仔细阅读。

提交 Bug

在测试完成后，如果参赛者发现了 bug，则可以在 bug-hunting/issue 按照模版提交 Issue。

评估 Bug

PingCAP QA 团队会根据 Bug 等级，为 Bug 打上 label（P0，P1，P2，P3），如果你是第一个发现 bug 的小伙伴，官方会根据 bug 的等级给你加上积分

积分兑换

积分获得情况将会在此页面底部积分排行榜呈现。在比赛结束一年之内都可以兑换 PingCAP 设置的奖项。

PingCAP

Thank You!