



下载APP



09 | 为什么ECB模式不安全？

2020-12-14 范学雷

实用密码学

[进入课程 >](#)**讲述：范学雷**

时长 11:07 大小 10.20M



你好，我是范学雷。

上一讲，我们讨论了对称密钥分组算法的计算过程，我们找到了影响对称密钥算法安全性的五个关键因素，以及初始化向量对算法安全性的影响和选择。

不过，还有一些遗留的问题，我们没有来得及讨论，链接模式和数据补齐方案对算法安全性有什么样的影响？它们是怎么影响分组算法安全性的呢？我们又该怎么避免这些安全陷阱呢？



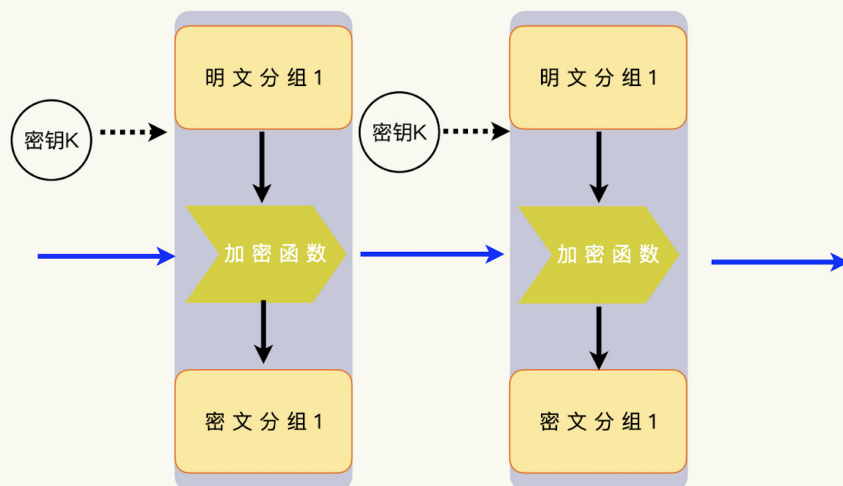
其实，这都是对称密钥分析的核心问题。因为，可以说，每一种链接模式、每一种数据补齐方案都有着不同的构造，当然也就对应着不同的分析办法，而且分析起来都较为复杂。

这一讲，我们先来分析链接模式对安全性的影响，同时，我们还可以借此机会研究一下 ECB 模式到底有什么问题。还记得吧？我们在开篇词提到过，它不是一个安全的加密模式。

在讲 ECB 模式之前，首先，我们先来看看链接模式是怎么一回事。

链接模式怎么连？

我们上一讲说过，链接模式指的是如何把上一个分组运算和下一个分组运算联系起来，使得上一个分组运算可以影响下一个运算。但是，这个联系是怎么建立起来的，上一个运算到底又是怎么影响下一个运算的，这个描述是模糊的。



从道理上来说，上一个分组运算的所有要素，都有可能参与到下一个分组运算里；下一个分组运算的每一个要素，都有可能接收上一个运算的一个要素或者几个要素的组合。

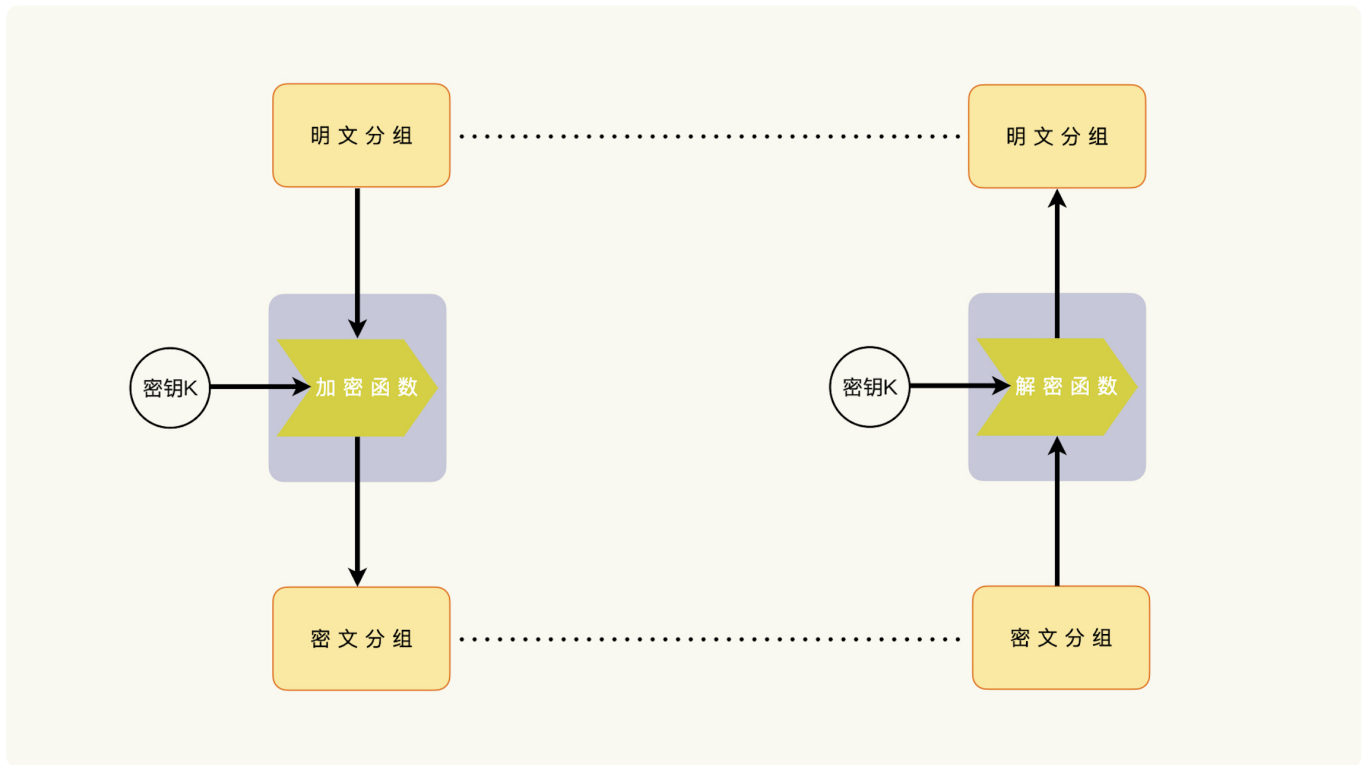
而在这之间，就会形成不同的分配组合，也就形成了不同的链接模式。

当然，你要知道，不是所有的组合都是安全的。其中，有些模式有严格的适用范围。超越了这个范围，这个算法就不再安全了。

我们需要特别关注这些限制，把每个模式的原理理解清楚，使用时不要掉进安全陷阱里。

ECB 模式什么样？

我们先来简单地了解一下 ECB 模式是怎么工作的。



上一次，我们讨论了影响对称密钥算法安全性的五个关键因素。**但是，ECB 模式有所不同，它不使用链接模式，因此它也用不着初始化向量。**

不使用链接模式，就意味着上一个分组运算不影响下一个分组运算，每一个数据分组的运算都是独立的。不需要初始化向量、每一个数据分组运算都是独立的，这特性令人振奋。


不需要初始化向量，还意味着 ECB 模式没有初始化向量管理的烦恼，有着更简单的代码。很多密码算法的接口设计和实现，为了方便使用，会使用缺省的数据。

比如说，下面的 Java 代码，使用的就是缺省的 ECB 模式。这段代码，看起来还算清爽简单：

```
1 Cipher cipher = Cipher.getInstance("AES");  
2 cipher.init(Cipher.DECRYPT_MODE, secretKey);
```

复制代码

但是，你看，下面的代码，看起来就繁琐的多。因为我们要在加密端和解密端同步初始化向量，而且它实际上的实现要更繁琐：

 复制代码

```
1 IvParameterSpec ivParameters = new IvParameterSpec(ivBytes);  
2 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");  
3 cipher.init(Cipher.DECRYPT_MODE, secretKey, ivParameters);
```

每一个分组运算都是独立的，这更是一个充满诱惑的特性。因为，能够独立运算，也就意味着可以并行地运算，也不必按照次序运算。

可以并行运算，意味着运算效率的大幅度提高；没有次序要求，意味着每一段加密数据都可以独立地存取、解密、修改、删除。而且，添加、插入新的数据段，也不会对其他数据段产生影响。


所有的这些，都意味着良好的运算效率。这是每一个大型数据计算场景渴望的特性，比如说数据库、流媒体、视频会议。遗憾的是，ECB 模式有致命的安全缺陷。

它的致命的安全缺陷却恰好来源于它令人兴奋的特性：初始化向量的缺失和链接模式的缺失。这意味着，如果我们不能拒绝它的这些诱惑，去寻找使用替代的方案，我们就会掉进这些致命的安全陷阱。

缺失带来了什么问题？

你可能还不知道初始化向量和链接模式的缺失会带来什么问题，让我们来一起分析下。


还记得初始化向量的特点吗？如果一个加密运算缺失初始化向量，相同的明文分组就会被加密成相同的密文分组。让我们一起来看一个例子，假设有如下一段数据：

 复制代码

```
1 ABCDEFGHHIJKLMNOP00123456789012345
```


当我们使用 AES-128/ECB 算法加密时，我们需要把这段数据分组成两个明文分组：

```
1 ABCDEFGHHIJKLMNO
2 0123456789012345
```

 复制代码


然后，我们加密这段数据，得到的密文（密钥 “1234567890123456” ）是（十六进制表示）：

```
1 1389AE9853633EBF3D35F28987FCD1187B4BFC89DD1700154482BC7EB686BB0E
```

 复制代码

我们可以把密文按照块大小分成两段：


```
1 1389AE9853633EBF3D35F28987FCD118
2 7B4BFC89DD1700154482BC7EB686BB0E
```

 复制代码

根据 ECB 加密模式的特点，我们知道第一行密文对应的数据是 “ABCDEFGHHIJKLMNO” ，第二行密文对应的数据是 “0123456789012345” 。


如果我们知道数据块对应的密文，我们就可以通过寻找重复的密文，在没有密钥，也不执行解密操作的情况下，知道对应的数据块。比如下面的密文：

```
1 7B4BFC89DD1700154482BC7EB686BB0E1389AE9853633EBF3D35F28987FCD118
```

 复制代码

对应的数据块是：

```
1 0123456789012345ABCDEFGHHIJKLMNO
```

 复制代码

你可能会有疑问，数据都是加密的，攻击者怎么会事先知道数据块和对应的密文呢？其实，互联网世界里，已知重复的、位置确定的数据非常多，HTTP 的头部数据，HTTPS 数据包的头部，URL 等都是重复频率很高的数据。

只要通过定位数据块，锚定对应的密文，就可以利用已知数据块和密文寻找、推断未知数据了。

而且，很多应用的场景，尤其是互联网的应用场景，注入特定明文数据、获取对应密文信息的攻击也是轻而易举的事情。如果攻击者没有“0123456789012345”的密文信息，他可以构造一个这样的明文，然后让密钥持有者加密，然后他就可以获得对应的密文分组。

攻击者也许并不满足于看看数据、窥视隐私。所以，ECB 模式更严重的问题，是由链接模式的缺失带来的。链接模式的缺失，会让每一个分组运算都是独立的，密文数据当然也会是独立的。

我们一起来看下面这段数据：

```
1 张三于二零二零年八月二十二日向李四借款人民币三十圆整，立此为证。
```

[复制代码](#)

假设我们加密运算时，这段数据被分割成如下的四个分组：

```
1 张三于二零二零年
2 八月二十二日向李
3 四借款人民币三十
4 圆整，立此为证。
```

[复制代码](#)

还有下面这段数据：

```
1 王二于二零二零年八月二十二日向李四借款人民币三十亿四千五百万六千圆整，立此为证。
```

[复制代码](#)

假设在加密运算时，这段数据也被分割成如下的五个分组：

```
1 王二于二零二零年
2 八月二十二日向李
3 四借款人民币三十
```

[复制代码](#)

- 4 亿四千五百万六千
- 5 圆整，立此为证。

如果有了明文，还有了对应的密文，我们就可以重新编排密文数据，以获得想要的结果。

如果使用了 ECB，我们可以把“亿四千五百万六千”这段明文分组对应的密文分组删除，那么解密的结果就是：

- 1 王二于二零二零年八月二十二日向李四借款人民币三十圆整，立此为证。

[复制代码](#)

我们也可以把“亿四千五百万六千”这段明文分组对应的密文分组插入到第一段数据的密文数据里。那么，修改的密文数据解密结果就是：

- 1 张三于二零二零年八月二十二日向李四借款人民币三十亿四千五百万六千圆整，立此为证。

[复制代码](#)

这样的结果，和原始数据相差甚远。这种攻击方式，就是常说的“分组重放”攻击。

什么时候使用 ECB 模式？

现在，你知道了 ECB 模式是一个很不安的模式了吧，既然 ECB 模式这么不安全，为什么安全应用程序接口还要提供 ECB 模式的编程接口呢？这主要是因为，ECB 模式是分组算法的基础。

分组算法的最基本运算，就是转换一个固定长度的数据块。这个单个数据块的转换，就是 ECB 模式下的运算。有密码学专业知识的算法工程师，可以通过合理地使用 ECB 模式，来构造更复杂、更安全的算法。**不过，我们不应该在一般的应用程序使用 ECB 模式。**

ECB 从诞生开始，就被认为是一个不安全的加密模式，有密码学基础知识的工程师，一般不会使用它。下一节，我们会讨论一个被广泛使用的模式，以及它现在面临的挑战。

Take Away（今日收获）

今天，通过解构 ECB 模式，我们讨论了在分组运算里，链接模式和初始化向量在分组运算里承担的角色，以及初始化向量缺失和链接模式缺失带来的问题。

最后，我们还讨论了 ECB 模式的实际用途，以及一个实用性的建议：我们不应该在一般的应用程序使用 ECB 模式。

通过今天的讨论，我们要：

了解分组算法里，初始化向量缺失可能带来的问题；

了解分组算法里，链接模式缺失可能带来的问题；

知道 ECB 模式不是安全的模式，应用程序不应该直接使用 ECB 模式。

思考题

今天的思考题，可能是一个稍微需要撕扯一下的挑战题。

我们前面说过，我们不应该在一般的应用程序使用 ECB 模式。可是，如果需要加密的数据小于一个数据分组（也就是说，数据太短，不需要使用链接），那么，我们可以使用 ECB 模式吗？有什么好处？有没有风险？

欢迎在留言区留言，记录、讨论你的想法。

好的，今天就这样，我们下次再聊。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 加餐 | 密码学，心底的冷暖

精选留言 (4)

写留言



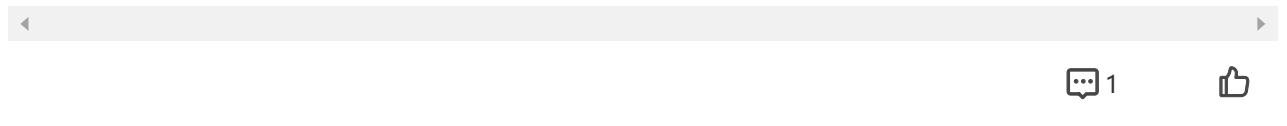
天天有吃的

2020-12-18

问题2：分组重放攻击的最小单位就是一个组吗？可不可能把某个组内内容修改了？或者再添加一个组，让原本的明文变长？

展开 ∨

作者回复：当然有可能修改组，这是别的攻击方式，就不是分组重放攻击了。



天天有吃的

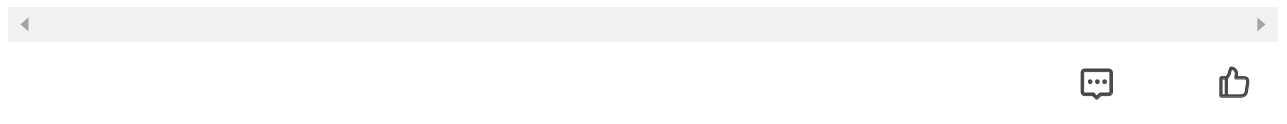
2020-12-18

小白打卡中...

问题1：王二于二零二零年八月二十二日向李四借款人民币三十亿四千五百万六千圆整，立此为证。这里有五行，为啥是四个分组？这里的四组是举例还是固定只能四组？

展开 ∨

作者回复：写错了，应该是5个分组。谢谢。



Ender0224

2020-12-16

好处：来自于ECB模式本身的好处，无初始化IV和串行计算，性能会提升。

风险消减点：

没有链接，那么就不会被实施“分组重放”攻击，即攻击者无法通过篡改信息影响系统可用性。

...

展开 ∨

作者回复：很棒！分组攻击这一点，还需要考虑更多的使用场景。





LittleQ4

个人看法，对于比较小的数据分组如果说每次传递的数据都不重复的话 可以使用ECB模式 好处是不需要使用iv数据长度比较短 风险就是 如果两次传递相同的内容 最终的加密结果也是相同的



1

