15 | AEAD有哪些安全陷阱?

2020-12-28 范学雷

实用密码学 进入课程 >

讲述：范学雷
时长 10:35 大小 9.71M

你好，我是范学雷。

上一讲，我们讨论了加密数据如何才能够自我验证，自我验证就是指解密的时候，还能够同时检验数据的完整性。我们还谈到了带关联数据的认证加密（AEAD）是目前市场的主流思路。

我们有了带关联的认证加密算法，应用程序再也不需要自行设计、解决数据的完整性问题了。但问题是，如果我们要在应用程序中使用带关联数据的认证加密，有哪些算法可以使用?

带关联数据的认证加密算法，有没有需要小心的安全陷阱？这是我们这一次需要解决的问题。

有哪些常见的算法？

还是老规矩，我们先来看看有哪些常见的算法。现在，常见的 AEAD 模式有三种：

GCM；

CCM；

Poly1305。

一般地，我们可以把带关联数据的认证加密看做一个加密模式，就像 CBC 模式一样，我们可以和前面提到的 AES 等加密算法进行组合。但 ChaCha20 和 Poly1305 通常组合在一起；Camellia 与 AES 通常和 GCM 以及 CCM 组合在一起。

由于 AEAD 模式相对较新，而 3DES/DES 等遗留或者退役算法又存在明显的安全缺陷，所以，我们一般不会使用遗留或者退役算法的 AEAD 模式。

如果我们重新整理一下，综合考虑加密算法和加密模式，那么，当前推荐使用的、有广泛支持的、风险最小的算法是：

AES/GCM；

ChaCha20/Poly1305；

AES/CCM。

如果你的应用程序使用上述三个算法里的任何一个，就可以规避掉我们前面讨论的很多风险。经过了十多次的讨论，不知道你有没有变得小心谨慎？**我们在使用任何算法之前，都要先存疑，后求证**。这时候，你应该要问了，那这三个算法有没有风险？

AEAD 有风险吗？

还记得吗？我们上面提到的 AEAD 算法，都需要使用初始化向量。虽然 ChaCha20/Poly1305 不属于分组密码的分类，但是这个算法也需要使用初始化向量。

我们前面特意讨论过初始化向量选择的一个原则：**在一个对称密钥的生命周期里，初始化向量不能重复**。所以，对于 AES/GCM 算法，同样的对称密钥，一定不要使用重复的初始化向量。否则的话，就存在安全漏洞。

你发现了，**初始化向量的重复问题，就是使用 AEAD 算法的最大风险，也是最难处理的风险。**没有密码学经验的开发者可能会忽视这个风险，相应的应用程序也可能因此存在严重的安全问题。

在前面讨论分组对称密码的时候，我们讨论过两个初始化向量选择的方案：使用随机数以及使用序列数。然而，这两个方案也是各有各的缺陷。

单相思的随机数

我们先来看看，使用随机数作为初始化向量有什么问题呢？

谈到使用随机数作为初始化向量，我们指的是信息发送方的行为。既然信息的发送方使用的是随机数，信息的接收方怎么能知道这个随机数的数据呢？

不使用同样的初始化向量，信息接收方是没有办法解密加密数据的。**该怎么解决初始化向量在信息发送方和接收方之间的同步问题？这是使用随机数作为初始化向量必须要考虑清楚的问题。**

幸运的是，对于上述我们提到的 AEAD 算法，初始化向量并不需要保密。一个常用的方案，信息发送方发送加密信息的时候，可以把明文的初始化向量一起发送。信息接收方直接使用接收到的明文初始化向量，就可以解密加密数据了。

不过，随之而来的问题是，如果有攻击者修改了明文传递的初始化向量，信息接收方能够察觉吗？这个答案是肯定的。不同于没有验证的加密算法，AEAD 有自我验证的能力，如果初始化向量被篡改，解密过程中，数据验证这一关是通不过去的。

也就是说，解密不会成功，篡改可以被检测出来。

还有一个问题，我们需要注意随机数的大小。对于这个问题，我们要再次考虑初始化向量选择的原则，**在一个对称密钥的生命周期里，初始化向量不能重复。**也就是说，随机数必须有足够的容量，使得在对称密钥的生命周期里，都不太可能出现重复的随机数。

最常用的经验数据，是使用 64 位的随机数。64 位的初始化向量，支持 2^64 的加密运算。大部分的应用程序都难以达到这么大的运算量。

不过，需要注意的是，虽然随机数的文字里带有数字，我们也不能把随机数看做是整数。在密码学的范畴里，随机数指的是随机的数据，不是随机的数字。

所以，千万不要移除随机数头尾的零，要保持随机数完整的位数。64 位的随机数，就一定要有 64 位的数据。即使头尾可能是零，也不能掐头去尾变成 63 位或 62 位，或者画蛇添足变成 65 位。

使用随机数作为初始化向量，还有一个常被忽视的小缺陷。随机数的初始化通常是一个费时费力的过程。在应用程序里，大量的随机数实例的初始化可能会造成应用程序的性能问题，甚至包括应用程序的停顿。

不过，对于有经验的程序员来说，这是一个好解决的问题。只要选择好随机数发生器、减少随机数实例数就可以了。具体的情况，我们后面再讨论。

除此之外，使用随机数作为初始化向量，还有一个小小的缺陷，就是需要额外传递初始化向量。不过，对于大部分应用来说，这点小缺陷，并不是什么问题。

难断舍的序列数

在部分场景中，随机数的产生是一个费时费力的过程。但是，对于 AEAD 算法，只要初始化向量不重复就行，并不要求初始化向量不可预测。避免昂贵的随机数，是我们在使用密码学技术时，要经常考虑的问题。**使用序列数作为初始化向量，就是一个最流行的方案。**

如果使用序列数，你需要注意的是，使用静态的对称密钥，特别需要注意序列数的静态化。比如说，对称密钥存在磁盘上，每次启动程序，都加载该密钥。那么，如果序列数没有对应的保存下来，每次启动的应用程序就有可能使用重复的序列数，从而带来严重的安全问题。

如果序列数也保存在磁盘上，并且每次程序启动时，加载密钥的时候，也加载序列数，我们也需要注意多线程的同步问题，其实也不省心。

使用静态的对称密钥的问题很多，我们后面还要讨论其他的安全问题。根本的办法，就是避免使用静态的密码。对称密钥，最好使用的时候再生成它，使用完立即销毁，不保存，不长期使用（不超过 7 天）。

理想地，如果信息发送方的每一个信息，信息接收方都能够收到，中间没有攻击者的干扰，使用序列数的初始化向量就不需要传递。信息发送方和接收方各自独立维护一个序列数计数器，就可以保持初始化向量的同步问题了。

但是，这种状况太理想了，有很强的局限性，只能在严格的、没有序列数错配的场景使用。

在网络环境中，信息可能丢失，或者被丢失，或者被攻击者插入攻击信息，或者被攻击者丢弃部分数据，这些都会引起序列数的错配问题，打乱信息接收方的节奏，错乱信息接收方的计数器。

怎么能够获得序列数开销小的好处，同时避免序列数错配的问题呢？

序列数的开销小，直接受益方是信息发送方；

序列数错配，直接受害方是信息接受者。

如果信息接收方不需要保持序列数的状态，不需要匹配序列数，也就没有序列数错配的烦恼了。

其实，答案也很明显。信息发送方可以像使用随机数初始化向量一样，发送加密信息的时候，把明文的初始化向量（也就是序列数）一起发送。信息接收方直接使用接收到的明文初始化向量，而不用理会这个初始化向量是一个随机数，还是一个序列数，或者是其他的什么数据。

传送明文的序列数这一个方案，兼顾了信息发送方的效率，规避了信息接收方的顾虑，是目前最常用的初始化向量选择方案。

不过，还是要说一点注意的，既然信息接收方不再需要判别接收到初始化向量是一个随机数还是一个序列数，信息发送方就要兼顾到信息接收方的处理便利。

即使序列数是 1，也要按照固定的位数传输，比如我们前面讨论过的固定的 64 位数据。如果序列数较小，可以前面补零，直到满足 64 位的数据要求。

到目前为止，我们已经知道了，新的应用程序里，应该优先使用 AEAD 算法。AEAD 算法需要初始化向量，明文信息，关联信息，还有密钥。

你有没有想过这样的问题：密钥从哪里来的？密钥应该是什么样的？回答这些问题之前，我们要先讨论一个对称密钥依赖的技术，随机数。我们下次聊聊随机数以及随机数发生器。

Take Away（今日收获）

今天，我们讨论了常见的带关联数据的认证加密算法，以及使用带关联数据的认证加密算法的常见问题，也就是初始化向量的问题。

我们还列出了当前推荐使用的、有广泛支持的、风险最小的算法。它们是：

AES/GCM；

ChaCha20/Poly1305；

AES/CCM。

通过今天的讨论，我们要：

知道常用的三个 AEAD 算法；

知道 AEAD 算法初始化向量不能重复的要求；

知道 AEAD 算法常用的初始化向量选择方案。

思考题

今天的思考题，是一个需要动手的题。

在不同的章节，我们花了很大的精力来讨论初始化向量的问题。这是一个特别容易忽视的环节。

在你正在开发的项目中，或者你关注的开放源代码项目中，试着搜索、统计一下初始化向量的使用状况。加密端有没有使用重复的初始化向量？解密端有没有办法检验初始化向量的重复？初始化向量选择的是随机数还是序列数，或者随机数和序列数的组合？

你有没有发现不恰当的初始化向量使用方案？

欢迎在留言区留言，记录、讨论你的发现。

好的，今天就这样，我们下次再聊。

提建议

上一篇 14 | 加密数据能够自我验证吗？

下一篇 16 | 为什么说随机数都是骗人的？

精选留言 (3)

写留言

Litt1eQ
2020-12-28

想到了802.11当中的wep算法采用了24位的初始化向量与同一把密钥进行运算 从而导致了严重的安全问题

1

韩露
2021-01-06

AEAD 模式能跟国密算法结合吗？现在都在提倡国产化，如果不能跟国密算法结合，还是用不了AEAD模式。

展开

作者回复: 可以的

可怜大灰狼
2020-12-28

1.密钥固定不变
2.密钥定期变化，但是没用初始化向量
3.用上了初始化向量，但是这个向量是固定值
4.没用MAC，没用AEAD
完了，完了，该犯的问题，都有了。

展开

作者回复: 哈哈，能发现问题，就很好！