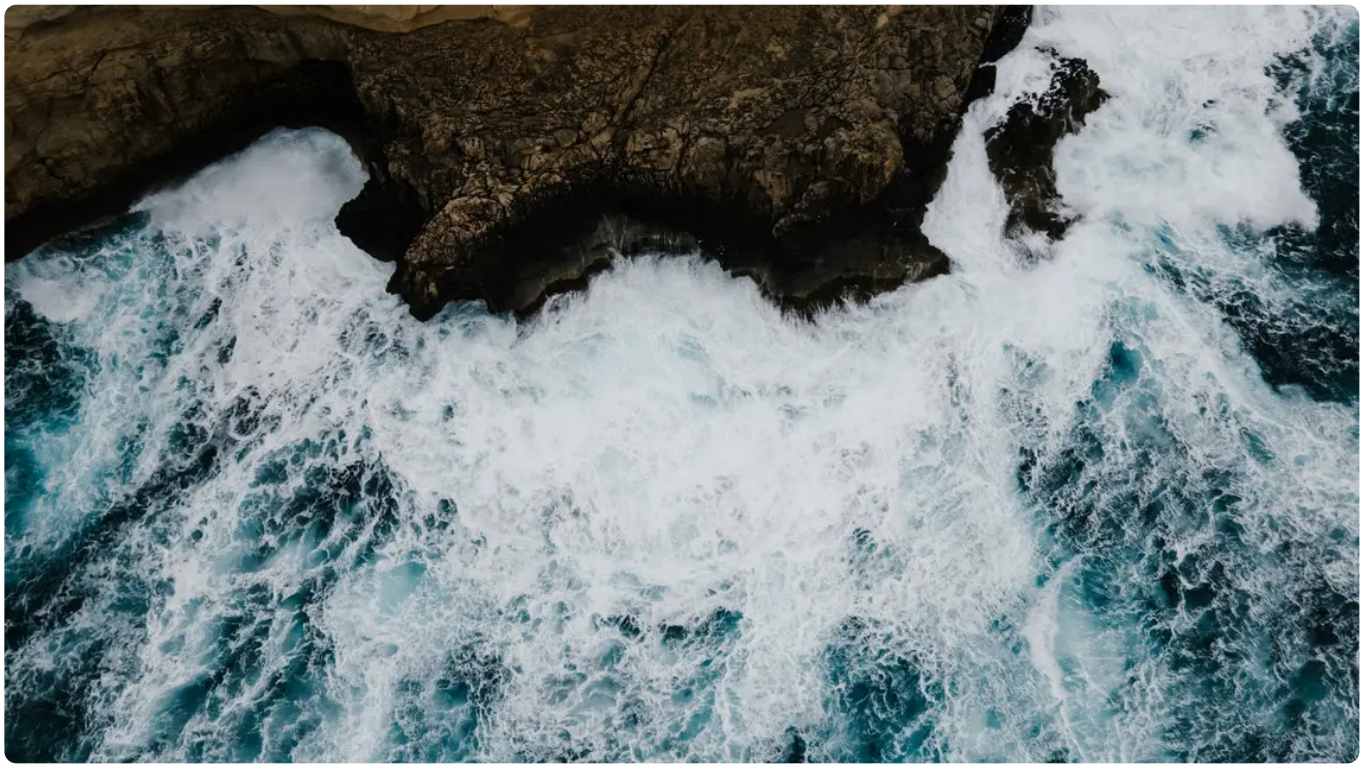
15 | 现代密码：你用的加密算法过时了吗？

2021-12-20 范学雷

《深入剖析Java新特性》 课程介绍 >



讲述：范学雷

时长 10:27 大小 9.58M

你好，我是范学雷。今天，我想和你聊聊 JDK 里的密码学算法相关的问题。

Java 语言安全的基础，主要有两块内容。一块是 Java 语言的安全设计，比如字节码的校验，内存保护机制等等；另外一块是 Java 平台的保护机制，比如签名的类库，资源的认证授权等等。而 Java 平台的保护机制，是建立在密码学的基础之上的。

这一次的讨论，我们从故事开始，来看看现在我们应该采用的密码学的技术，以及应该抛弃的密码学技术。

阅读案例

1976 年，是现代密码学的奠基之年。这一年，Diffie–Hellman 密钥交换协议公开发表。这是由 Ralph Merkle 构思并以 Whitfield Diffie 和 Martin Hellman 命名的第一个公钥协议。这是最早为公众所知的，提出公钥和私钥思想的著作。从这一年开始，在非安全通道上建立安全通信的想法，有了理论上的依据；现代互联网的安全，也终于有了稳固的基石。

Diffie–Hellman 密钥交换协议的论文，为密码学家展示了一个全新的大陆。有了这个方向的指引，接下来很快就有了更多的脚步踏出了新的道路。1977 年，受 Diffie–Hellman 密钥交换协议的启发，Ron Rivest、Adi Shamir 和 Leonard Adleman 公开发表了基于公开密钥的电子签名算法，也就是 RSA 算法。从此以后，要在非安全通道上识别身份、建立信任的想法，也有了理论上的依据。

至此，加上传统的加密技术，解决信息安全基本问题的三大技术就已经集结完成了。随着互联网的发展，这些技术大放异彩，成为了互联网基础设施最终的环节之一。

后面的事情也就顺理成章了。1982 年，Ron Rivest、Adi Shamir 和 Leonard Adleman 成立了 RSA 公司，公司主要提供基于 RSA 算法的产品和服务。1991 年，RSA 公司推出了 RSA 大会以及 RSA 算法的分解挑战。 2006 年 RSA 公司被 EMC 收购，收购价达到 21 亿美元。2007 年，RSA 算法分解挑战终止。而 RSA 大会，则发展成了信息安全领域最富盛名的的大会。

为什么 RSA 分解挑战终止了呢？ 按照官方的声明，因为：“现在业界对常见对称密钥和公钥算法的密码分析强度有了更深入的了解，这些挑战不再活跃。”

那分解挑战的成果是什么样子的呢？ 我想你也一定感兴趣。

1991 年 3 月 18 日，RSA 公司推出 RSA 算法的分解挑战。不到两个星期，也就是 1991 年的 4 月 1 日，330 位的 RSA 密钥被破解。随后，更高强度的 RSA 算法被破解。其中，768 位的 RSA 密钥在 2009 年被破解，这是一个 RSA 命运的分水岭。从此以后，小于或者等于 1024 位的 RSA 密钥，都被认为是不安全的密钥。现在的 RSA 算法，应该是用至少 2048 位的密钥。

对 RSA 算法的破解研究，并不仅仅局限于因式分解这样的纯计算游戏。比如说，早在 1998 年，就有密码学家发现了对 RSA 算法进行旁路攻击的办法。现在，如果是用测时攻

击（timing attacks），对于 1024 位的密钥，破解传统的 RSA 实现也就是分分钟钟的事情。

虽然，我们可以通过复杂的 RSA 实现来化解这样的攻击。但是，复杂的实现，意味着性能的损失以及维护的困难。到这里，我们已经可以依稀地听到 RSA 算法要告别历史舞台的声音了。

其实，任何一个密码学的算法，都有它的生命周期。从看似完美的问世，到实际破落的境地，也就是数十年的时间。

看向未来

但是，我们的隐私数据却需要上百年，甚至是永远的保护。有生命周期的算法，似乎满足不了这样的要求。密码学要始终看向未来。如果站在十年后看现在，我们怎么能保证万无一失呢？

十多年后，量子计算机大概率就能够问世了。而量子计算机的计算能力是非常恐怖的。现在我们常见的非对称密码算法所能提供的计算强度，在量子计算时代，也许就像是小孩子的玩具一样脆弱。所以，密码学家和各种组织都在紧锣密鼓地遴选“后量子时代”的非对称密码算法。

显然，我们不能等到“后量子时代”的非对称密码算法问世以后，再来保护我们的隐私数据。现在我们就需要这样的保护。而这其中最重要的方案，就是使用前向保密（Forward Secrecy）的安全协议。前向保密也就意味着，即使未来我们反复使用的密钥被破解，我们的数据依然能够得到保护。如果你想了解更多的关于前向保密的细节，请参考我在另外一个专栏里的讨论《量子时代，你准备好了吗？》。

在 Java 的设计和实现里，前向保密是 JDK 缺省的选择。这是 JDK 8 之后，JDK 做的一个重要的安全策略调整。这个调整，涉及到的大都是 JDK 实现的小细节，比如缺省 JDK 升级到 TLS 1.3 这样的变动。

JDK 的安全是 Java 语言的头等大事。所以，JDK 的安全改进一般情况下，都会向后移植，直到我们没有能力移植为止。前向保密的策略，也已经向后移植，进入到 JDK 8 了。

关注变化

既然密码学的算法有生命周期，我们就需要了解这个生命周期，及时地停止使用危险的、过期的算法。那么，哪些密码算法如今已经过期或者存在安全隐患？我们又能从哪里找到这方面最新的信息呢？

JDK 8 之后，Java 安全策略的另外一个重要的调整，就是公开发布 JDK 的密码路线图。在这个路线图里，JDK 会声明哪些密钥算法是危险的，哪些是过期的，以及 JDK 根据密码学的进展作出的变动。

如果你的产品或者代码涉及到了密码相关的内容，你就要密切关注这个路线图的更新，及时地调整产品里涉及到密码算法了。

另外，密钥算法的废弃，总是会带来这样或者那样的兼容性问题。当安全性和兼容性相遇的时候，我们应该毫不犹豫地选择选择安全性，及时解决掉兼容性问题。安全性问题，时间上千万不要拖。软件系统的漏洞，一般情况下，攻击者知道的比你还要早。我们拖拉的每一秒钟，都是留给攻击者的时间窗口。

应该抛弃的算法

下面我罗列了一些曾经流行的，JDK 支持的，但是我们不应该使用的密码学算法或者协议。继续使用这些算法，会给你的系统带来难以预料的灾难。而且，使用的系统也很容易成为黑客攻击的目标。

- MD2
- MD5
- SHA-1
- DES
- 3DES
- RC4
- SSL 3.0
- TLS 1.0
- TLS 1.1
- 密钥小于 1024 位的 RSA 算法

密钥小于 1024 位的 DSA 算法

密钥小于 1024 位的 Diffie-Hellman 算法

密钥小于 256 位的 EC 算法

应该退役的算法

下面我罗列了一些曾经流行的，JDK 支持的，我们可以使用，但是应该尽快替换掉的算法。这些算法，目前来看还是安全的，但是已经处于危险的边缘了。如果你的系统计划运行五年以上，这些算法的安全性值得担忧。

密钥大于 1024 位小于 2048 位的 RSA 算法。

密钥大于 1024 位小于 2048 位的 DSA 算法。

密钥大于 1024 位小于 2048 位的 Diffie-Hellman 算法。

RSA 签名算法

基于 RSA 的密钥交换算法

128 位的 AES 算法

推荐使用的算法

下面我罗列了一些现在流行的，JDK 支持的，我们推荐使用的密码学算法。这些算法，目前看还没有发现值得重视的安全问题，是可以信任的算法。如果一个系统计划运行五年以上，你应该使用这些算法。

256 位的 AES 算法

SHA-256、SHA-512 单向散列函数

RSASSA-PSS 签名算法

X25519/X448 密钥交换算法

EdDSA 签名算法

我们前面提到过，安全改进一般都会向后移植，但是也有我们没有能力移植的例子。上面提到的推荐使用的算法中， JDK 8 不支持 X25519/X488 密钥交换算法，也不支持 EdDSA 签名算法。一个最重要的原因，就是这些算法需要使用新的公开接口。

一般情况下，小版本的 JDK 升级，不能变更公开接口。这就让 JDK 8 有了安全上的短板。目前看，这个短板还不足以构成安全威胁。但是停留在 JDK 8 意味着我们放弃了更好的密码算法，包括安全性的提高和性能的提升。

我上面列举的算法，大部分开发者应该接触不到。因为，它们是 Java 语言和 Java 平台的一部分，是计算机基础设施的一部分。我们天天使用它们，但是没有多少人意识到它们的存在。如果你需要使用密码，比如签名 Java 包，或者使用数字证书，请留意这些数字内容使用的密码算法，尽量使用推荐的算法，千万不要使用已经抛弃的算法。

总结

好，到这里，我来做个小结。通过今天的讨论，我们知道，任何一个密码学的算法，都有它的生命周期。所以，我们要能够管理它们的生命周期。反映到代码里，就是要使用前向保密的安全协议以及当前推荐的算法；及时替换掉过期的算法。

对于 JDK 的开发者来说，我们要关注JDK 的密码路线图，了解 JDK 根据密码学的进展作出的变动，及时解决自己代码里的兼容性问题。

如果面试中聊到了密码学算法的问题，你可以聊聊前向保密，以及我们推荐的密码学算法。

思考题

今天的思考题，是一个拓展阅读。在上面推荐的算法里，除了 AES 算法之外，其他的三个算法，如果不是关注密码学进展的话，你可能都没有听说过。密码学进展很快，十多年前的主流算法，在今天几乎都要进入退休的年龄了。我们也要随时更新对密码学基本现状的认识。

如果有时间，你可以去搜索一下 RSASSA-PSS 签名算法，X25519 密钥交换算法以及 EdDSA 签名算法的相关介绍。不需要了解技术细节，知道大致是怎么回事就行。

欢迎你在留言区留言、讨论，分享你的阅读体验以及你的拓展阅读内容。我们下节课见！

赞 2 提建议

上一篇 14 | 禁止空指针，该怎么避免崩溃的空指针？

下一篇 16 | 改进的废弃，怎么避免使用废弃的特性？

精选留言 (3)

写留言

潜默闻雨
2021-12-24

量子计算机时代应该不能用传统加密思维了吧，非对称加密这种方式还是万变不离其宗，必须提升一个维度来适应时代的步伐，不然完全就是降维打击。

作者回复: 目前我还不了解有新维度的思维，密码学本质上是数学。

aoe
2021-12-21

已经做了笔记，保存了推荐使用的算法，下次使用时直接选老师推荐的，十分方便！

我还听说过：只要秘钥足够长，量子计算机也无法破解。一个科普工作者“科技袁人”也提到过：在量子时代，量子加密方会战胜破解一方，因为信息一旦被观察，加密方就会知道，这是由量子纠缠的物理特性决定的。

展开

作者回复: 没这么简单呢

kimoti
2021-12-20

老师,比特币的加密算法应该是很安全的吧,他的原理是基于椭圆曲线的,Java支持吗？

作者回复: Java支持椭圆曲线算法，椭圆曲线有很多变种的算法，安全性不能一概而论。