你好，我是李兵。

在《36 | HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

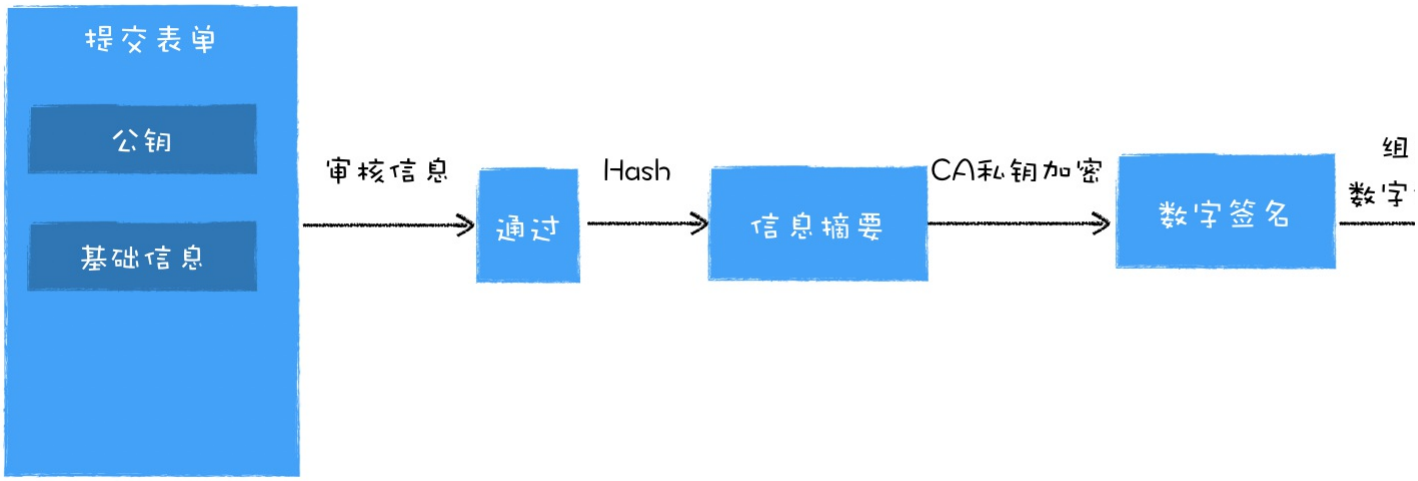
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

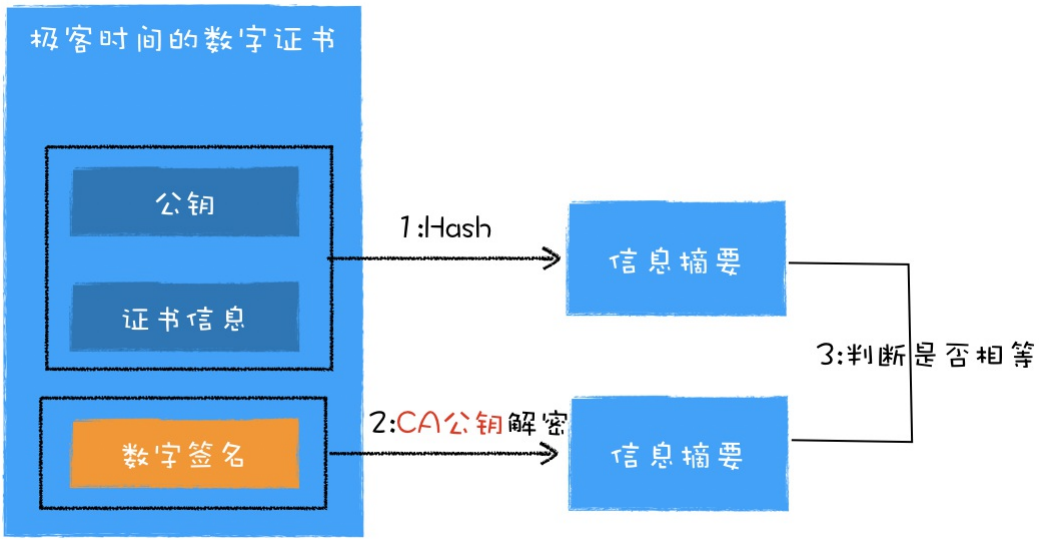
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

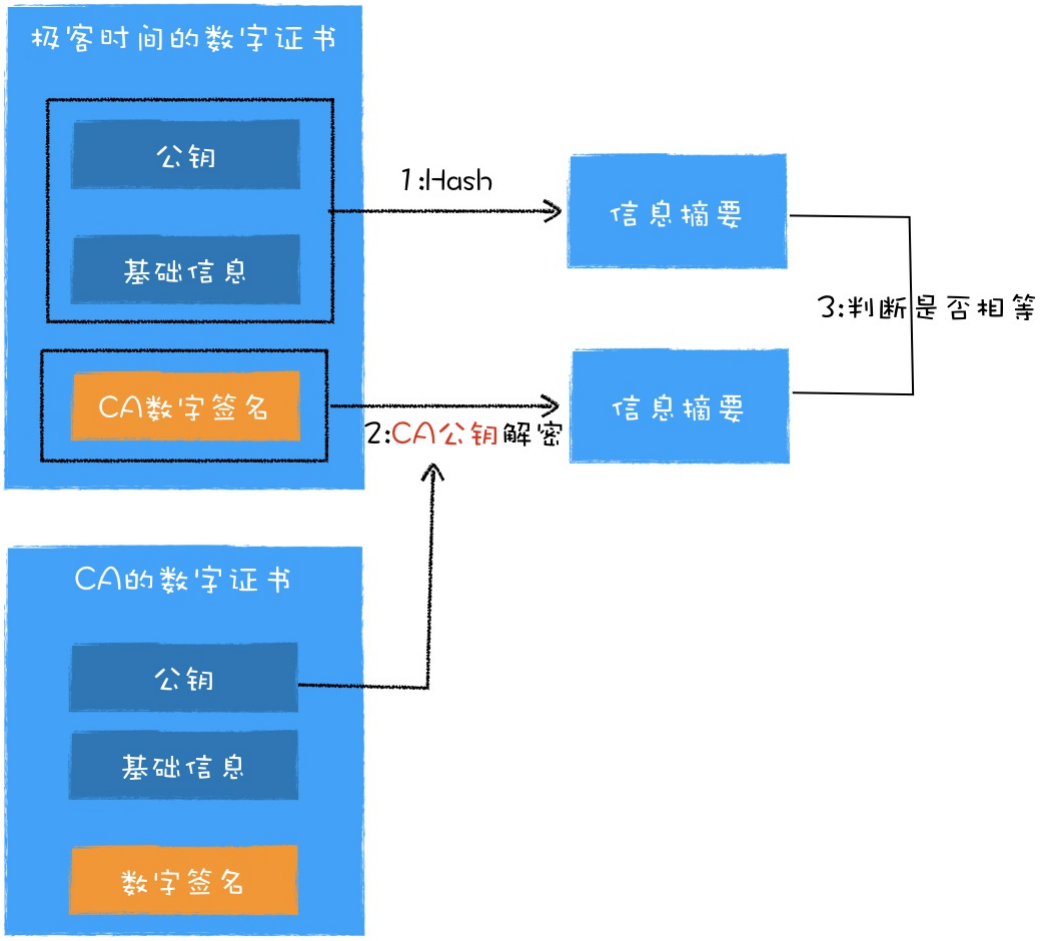
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



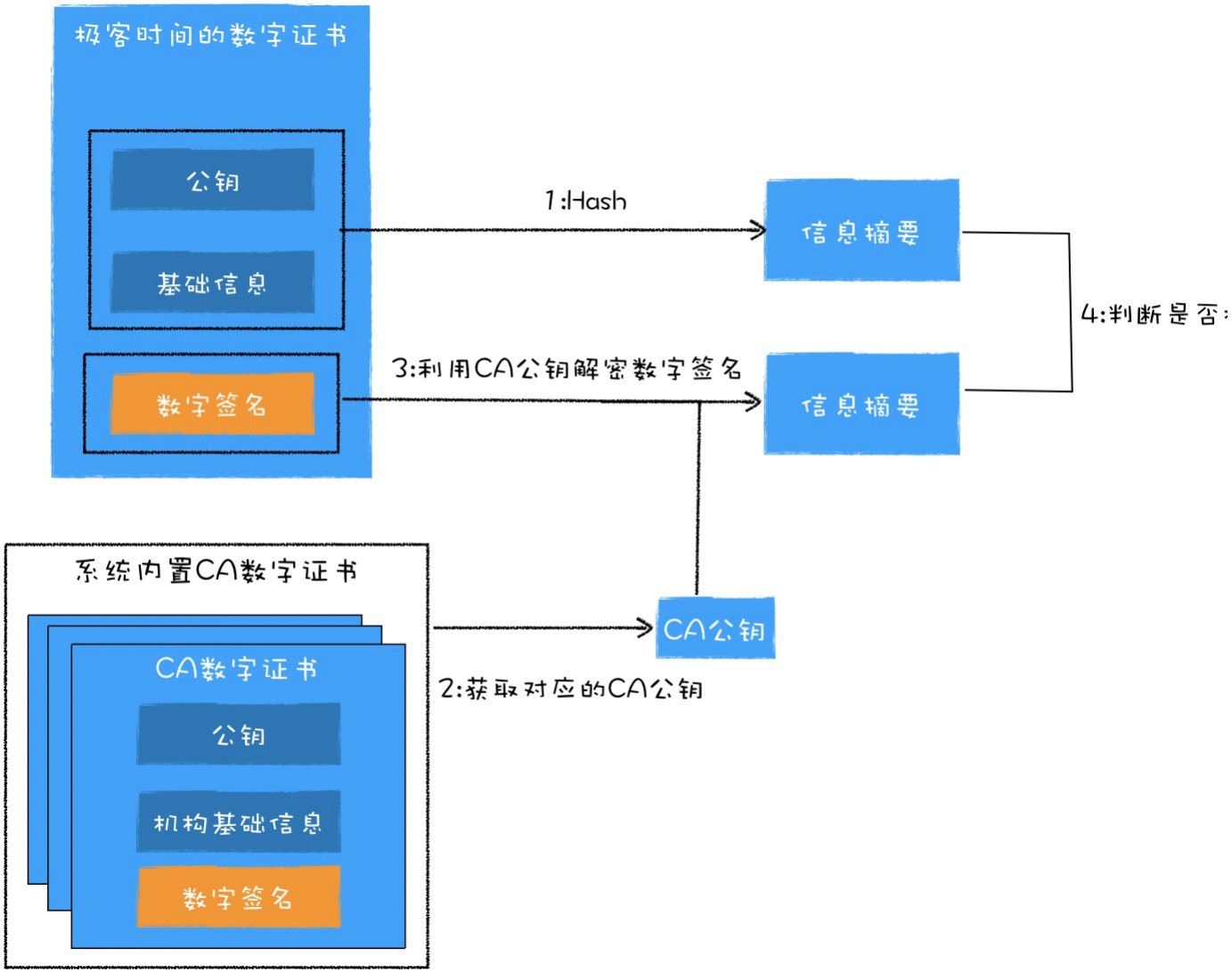
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

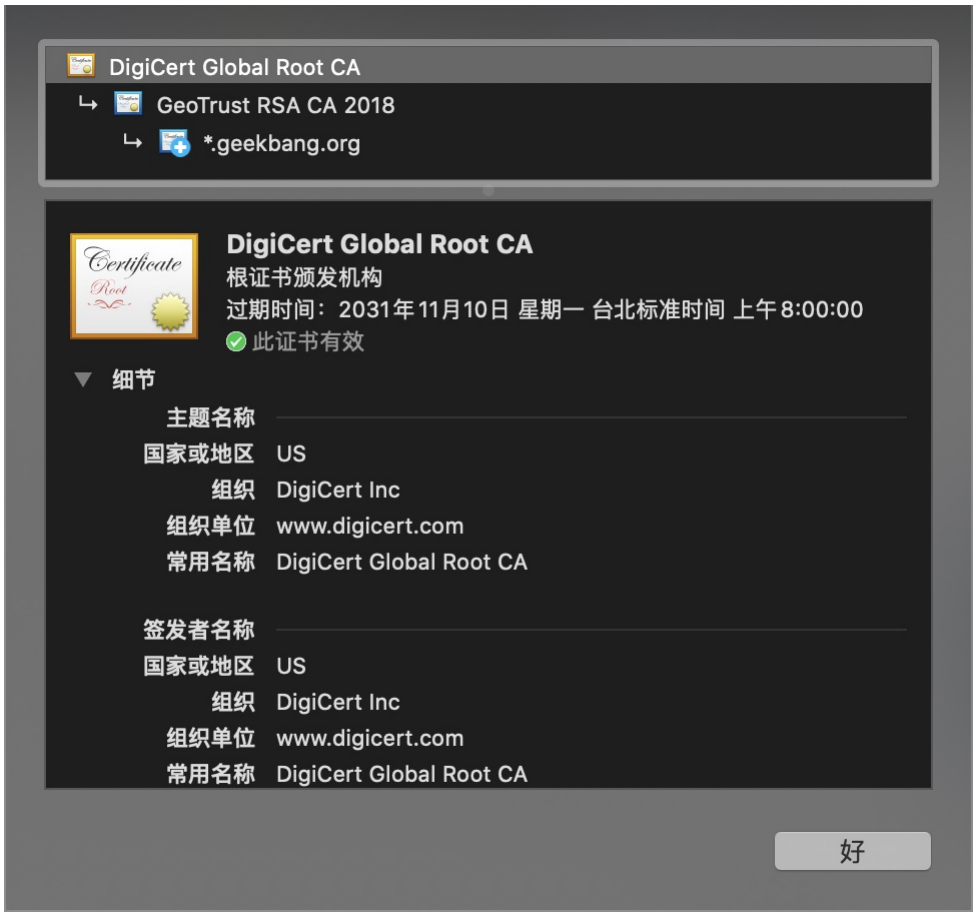
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

你好，我是李兵。

在《36丨HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

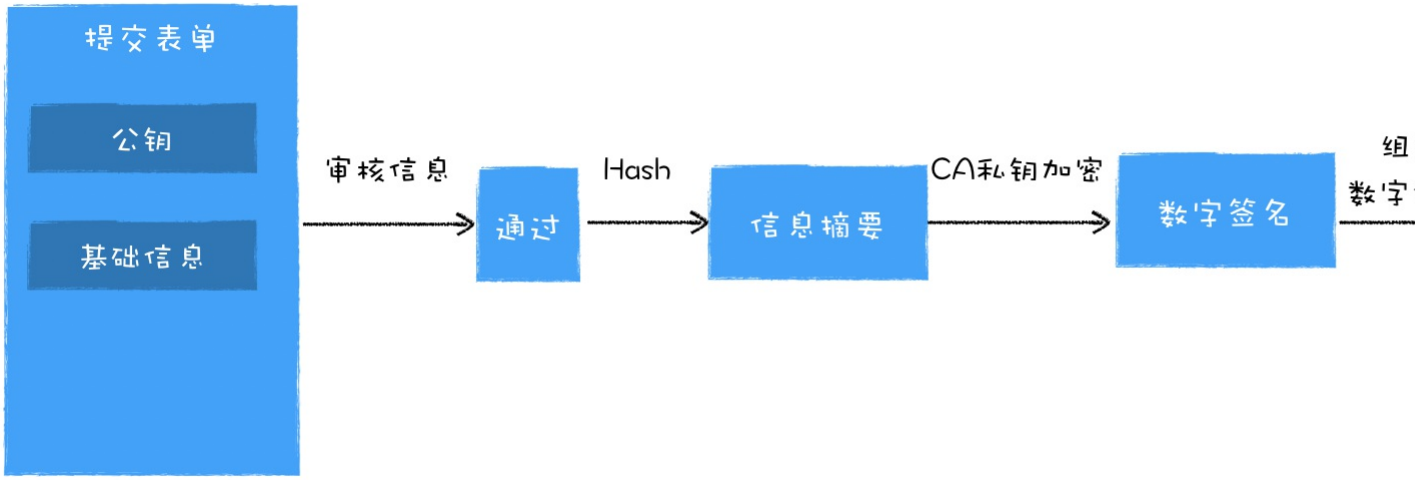
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

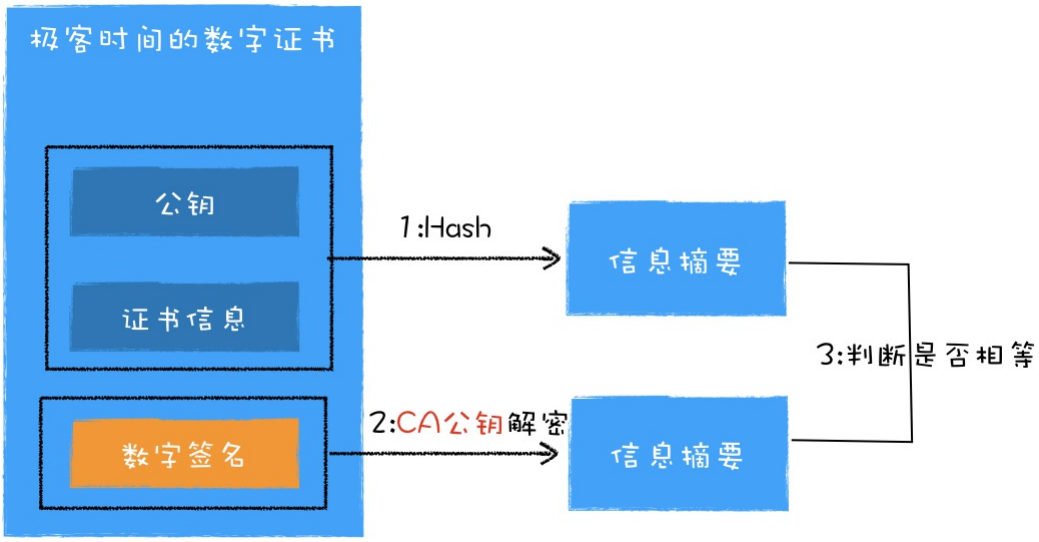
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

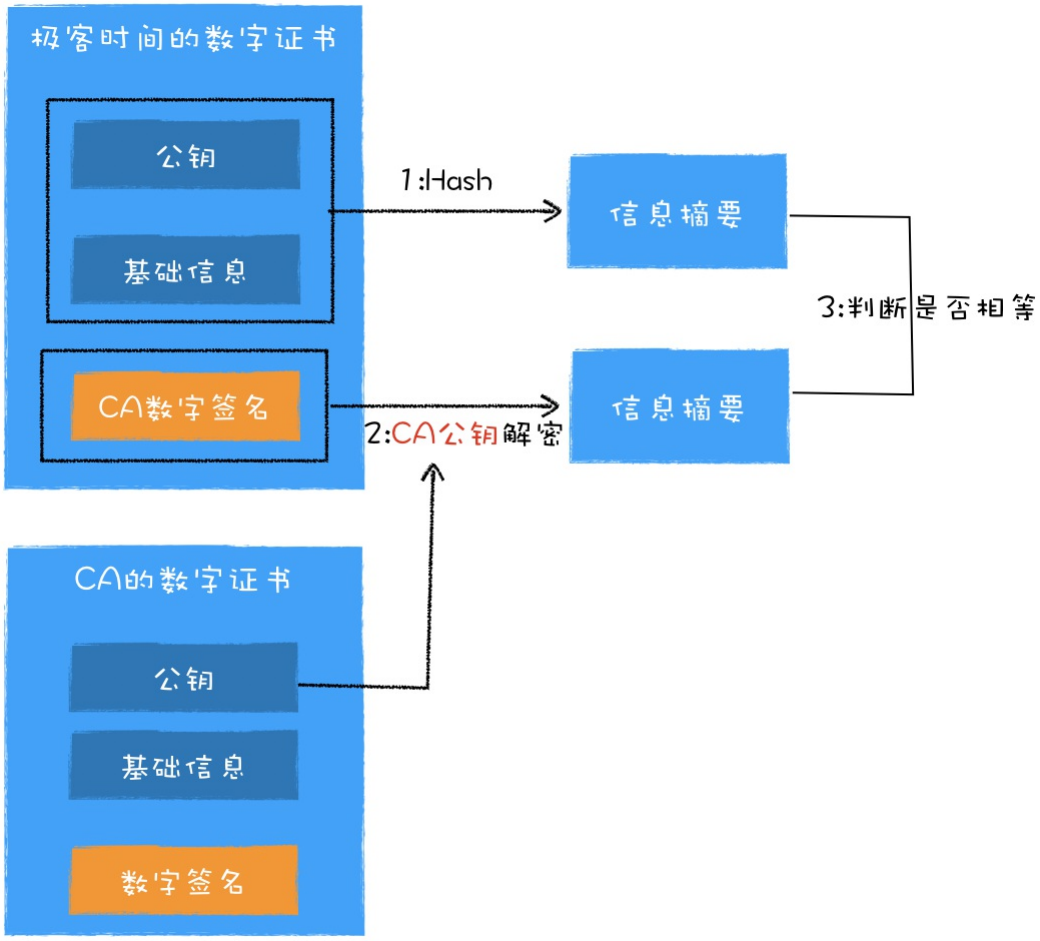
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



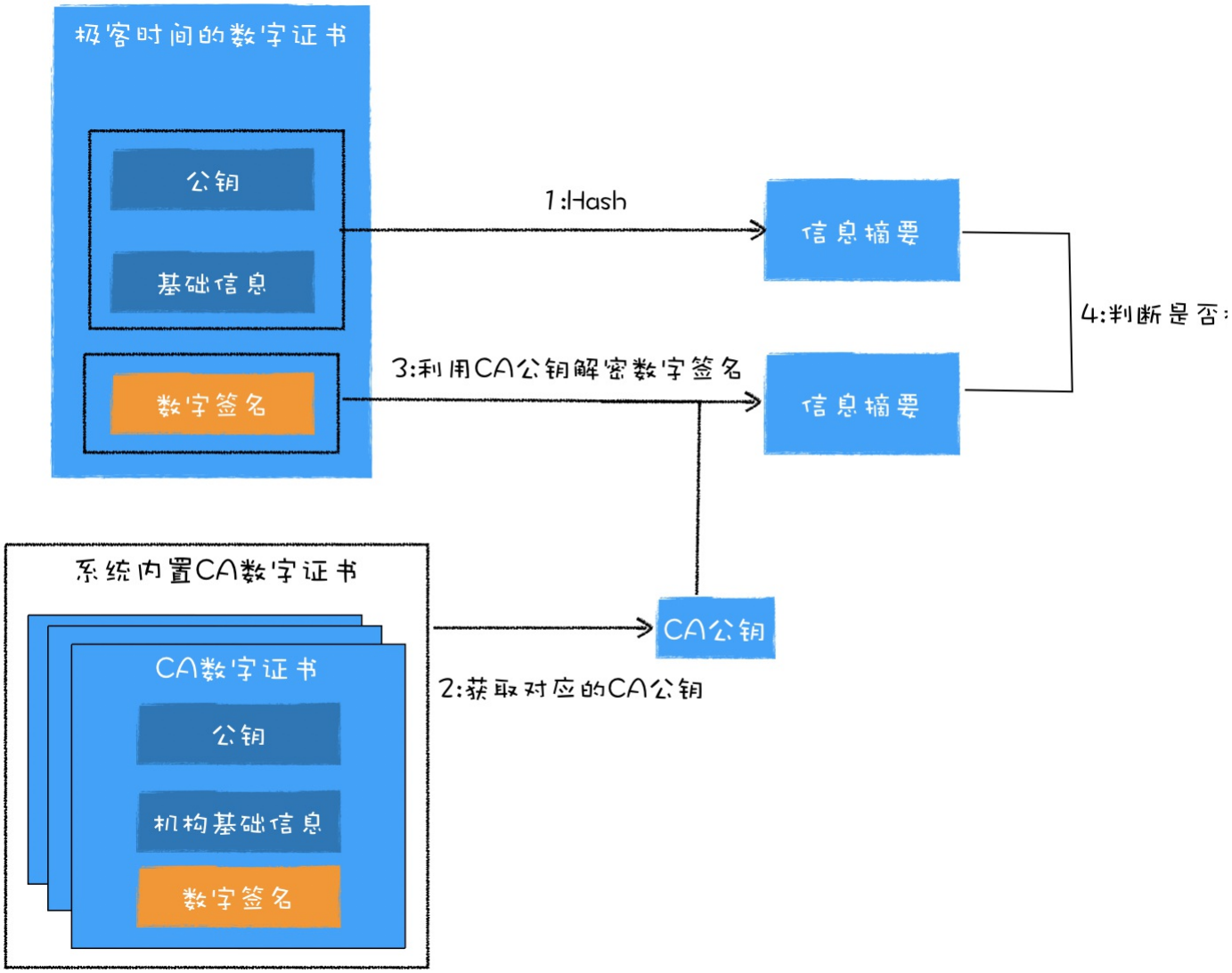
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

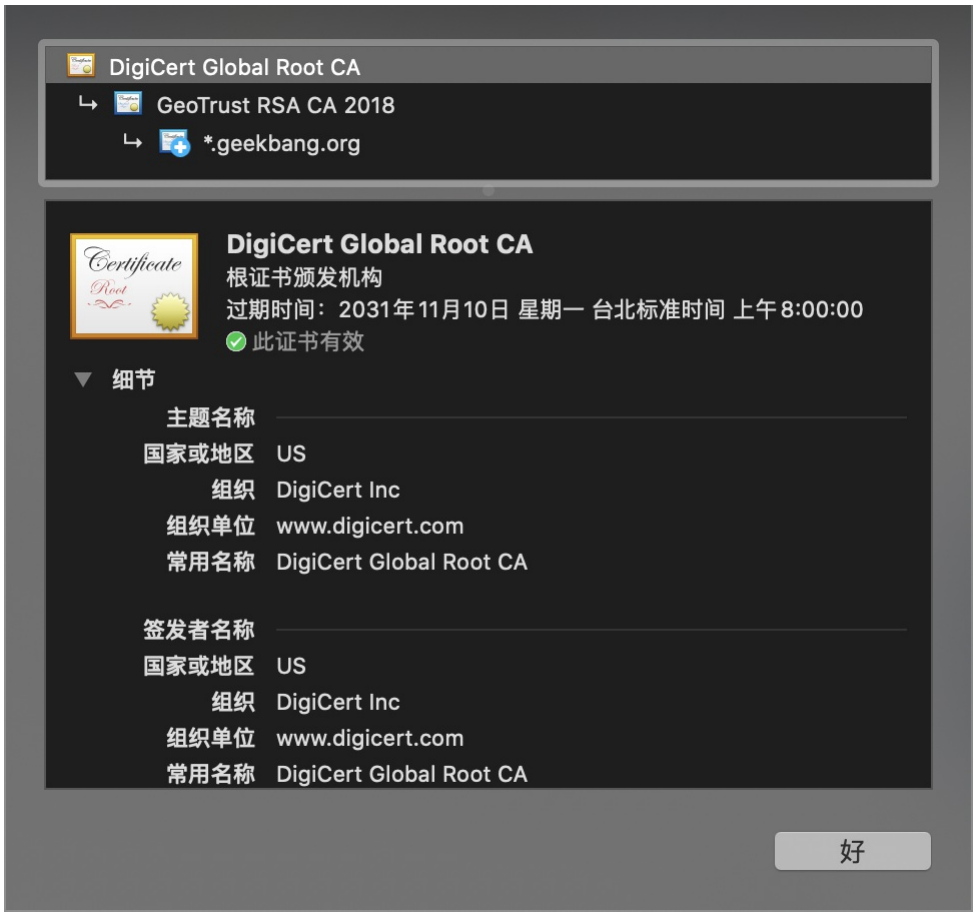
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

你好，我是李兵。

在《36丨HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

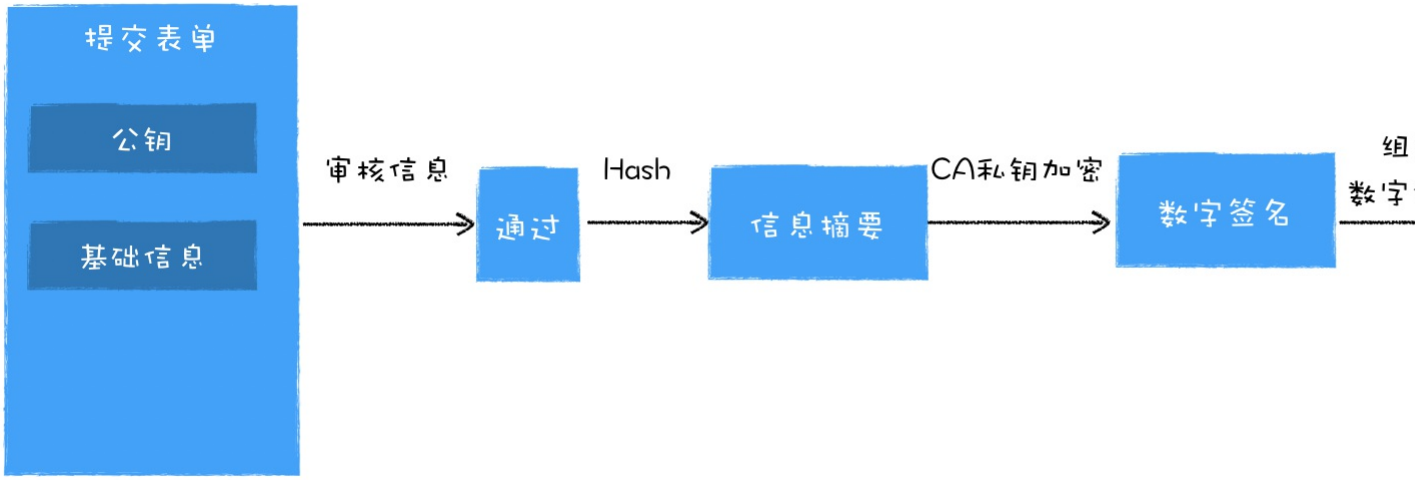
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

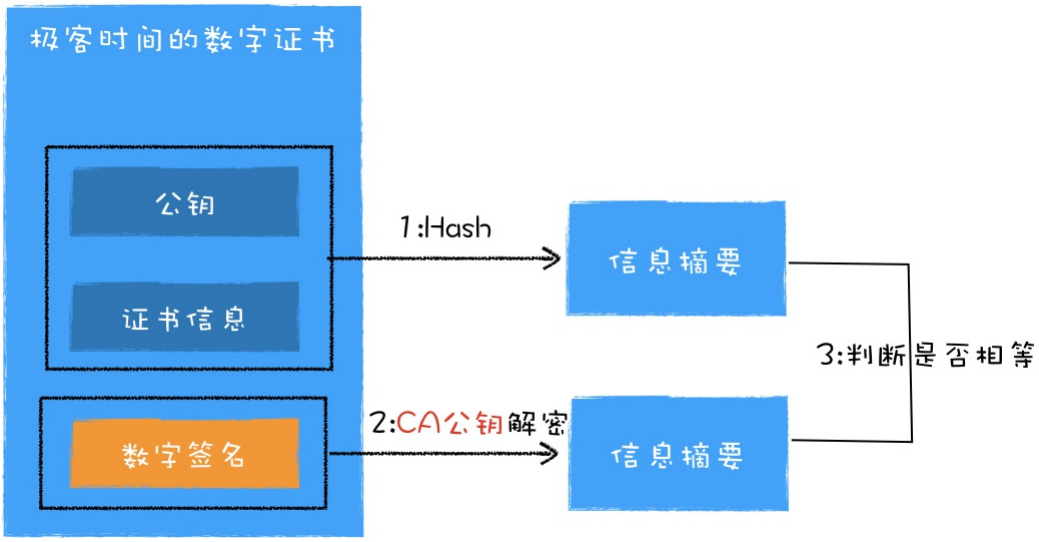
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

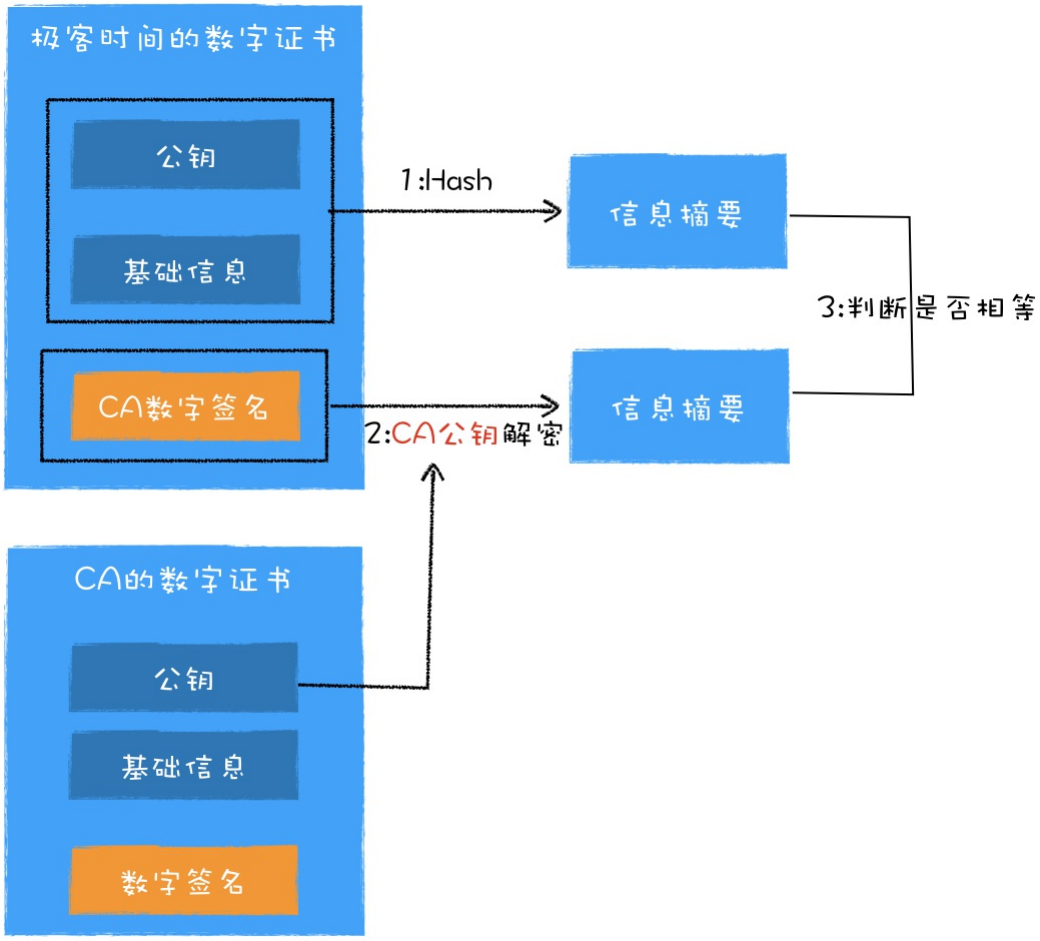
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



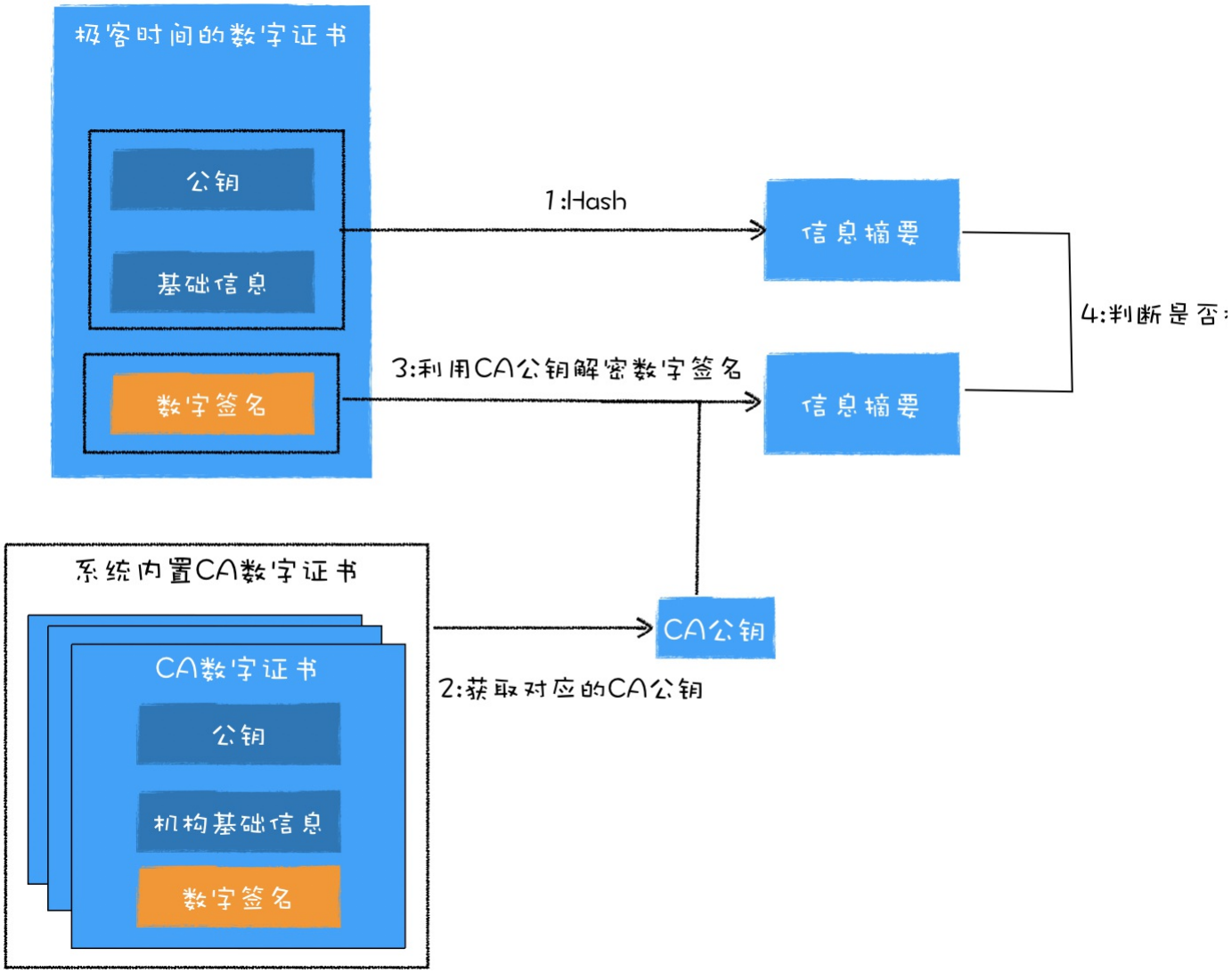
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

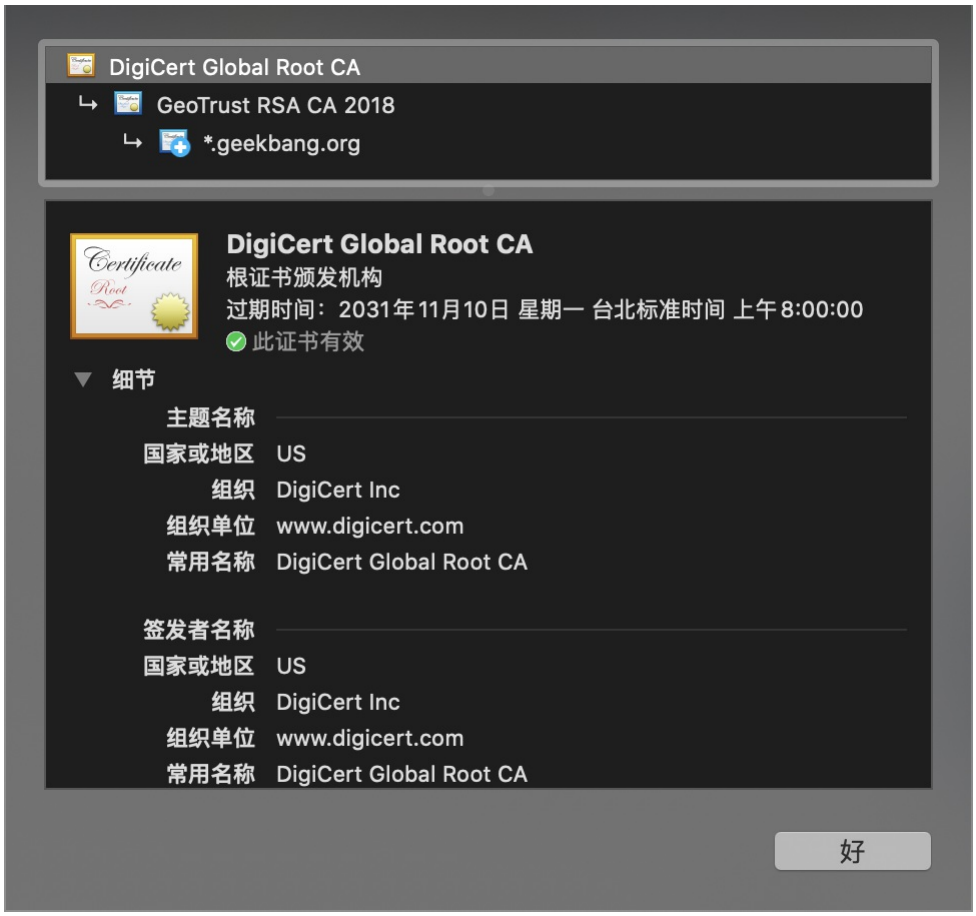
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

你好，我是李兵。

在《36丨HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

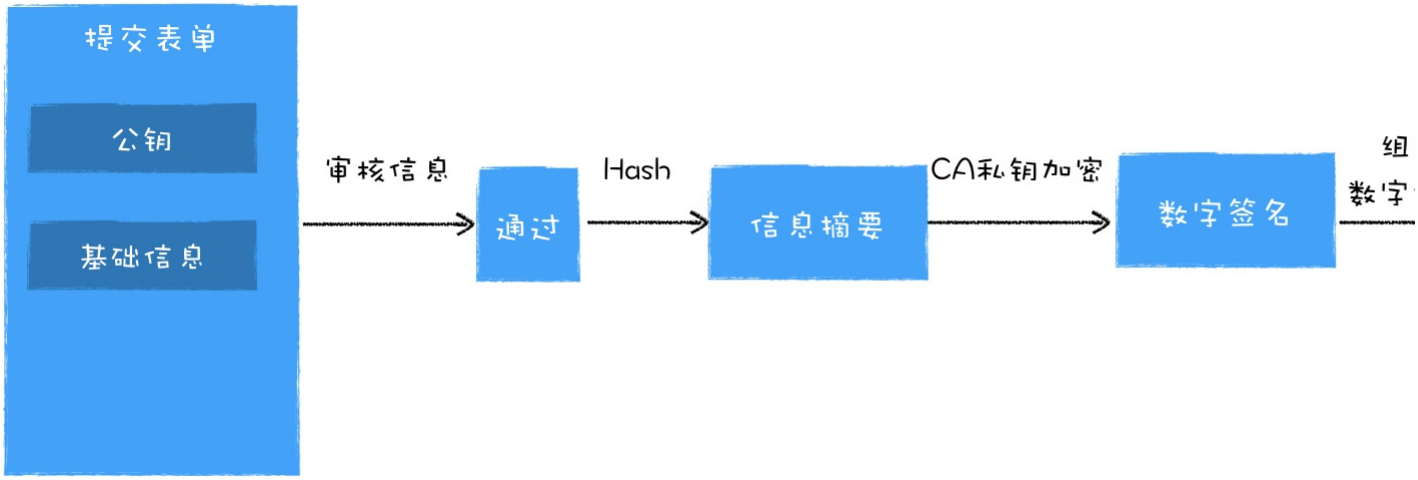
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

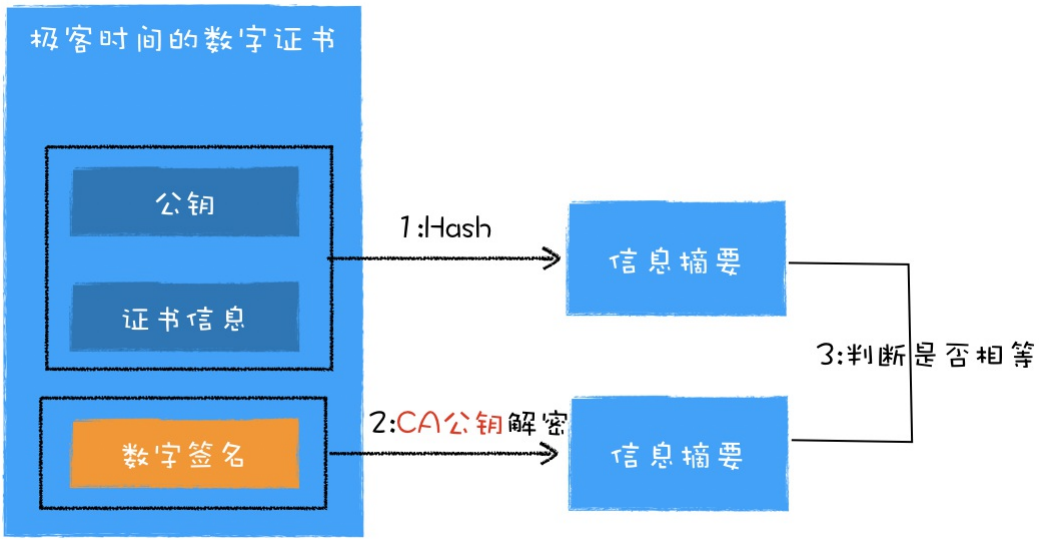
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

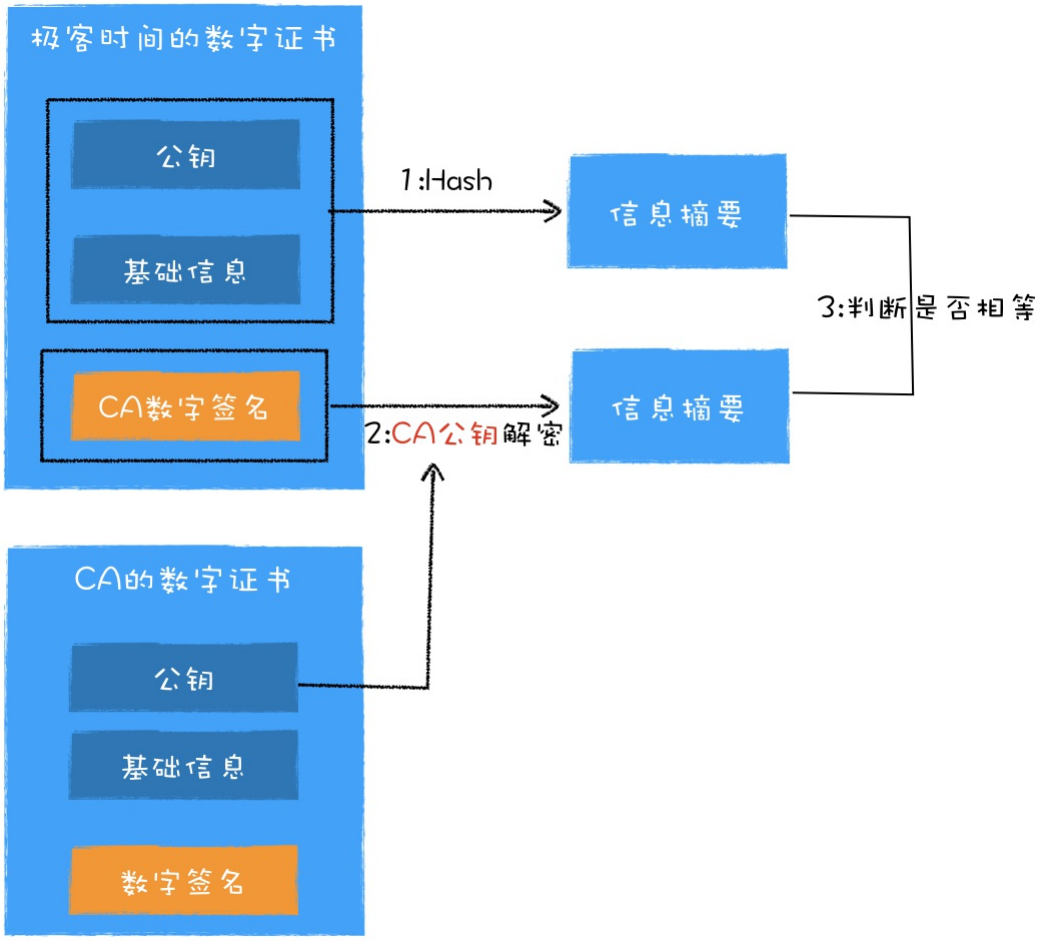
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



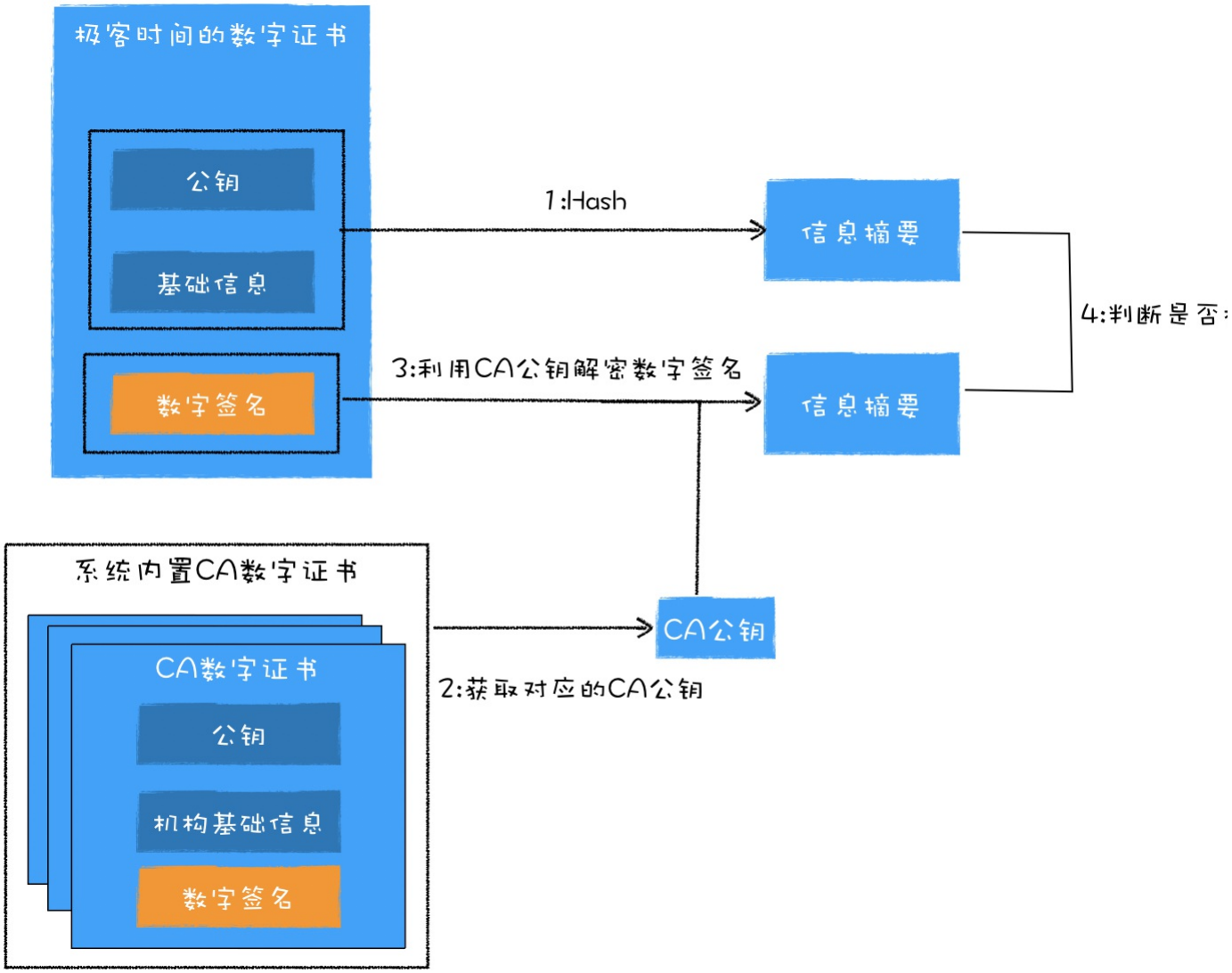
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

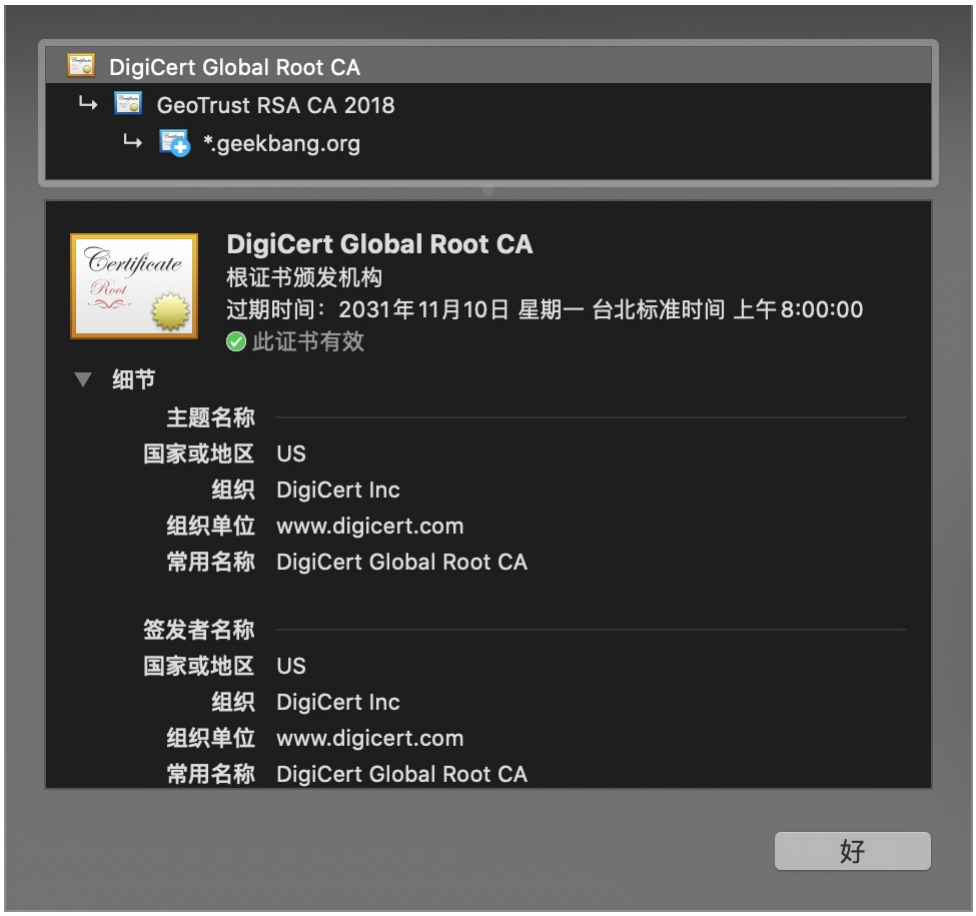
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

你好，我是李兵。

在《36丨HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

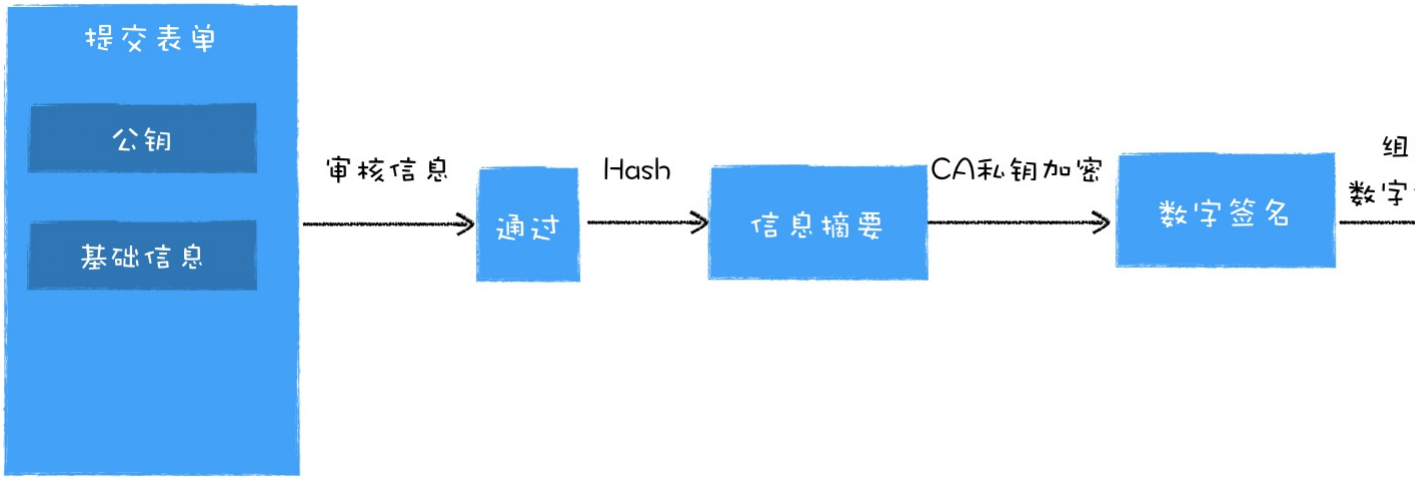
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

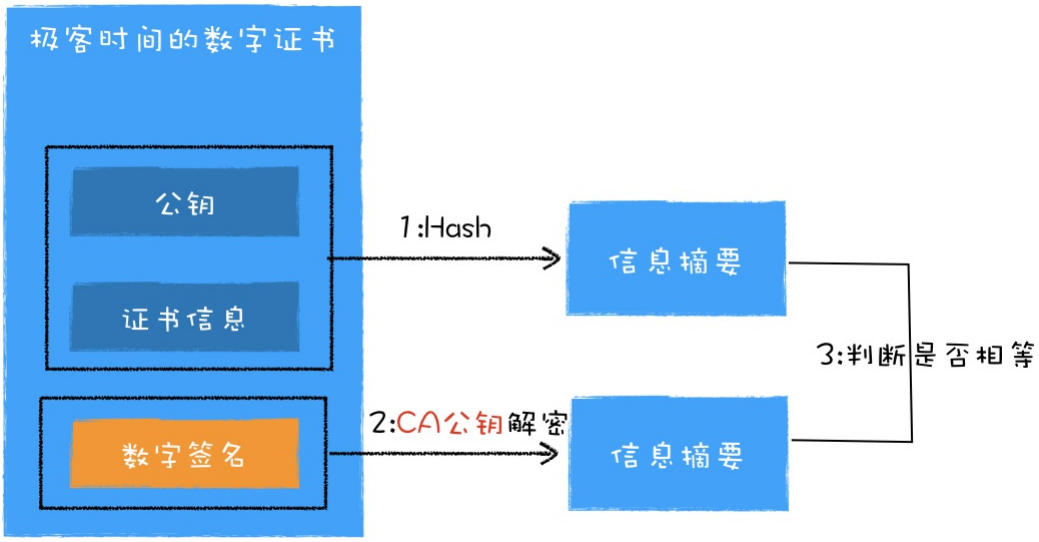
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

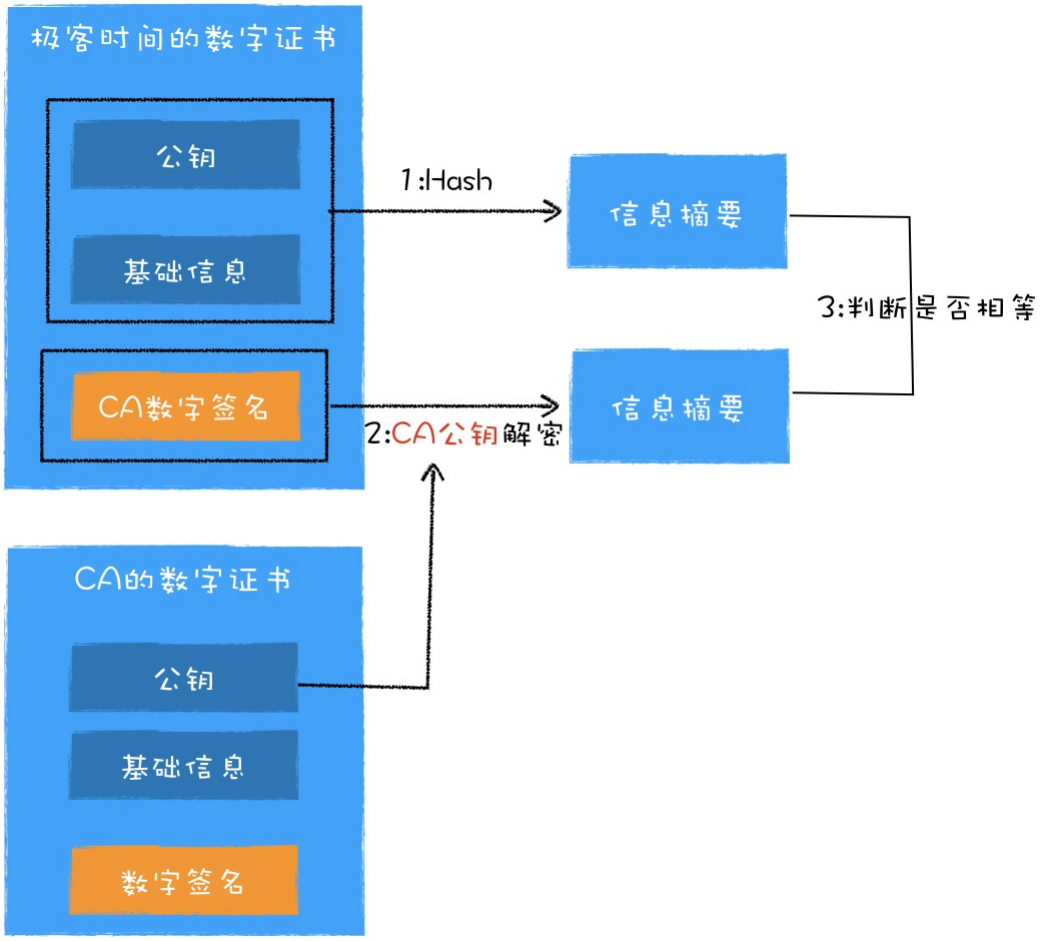
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



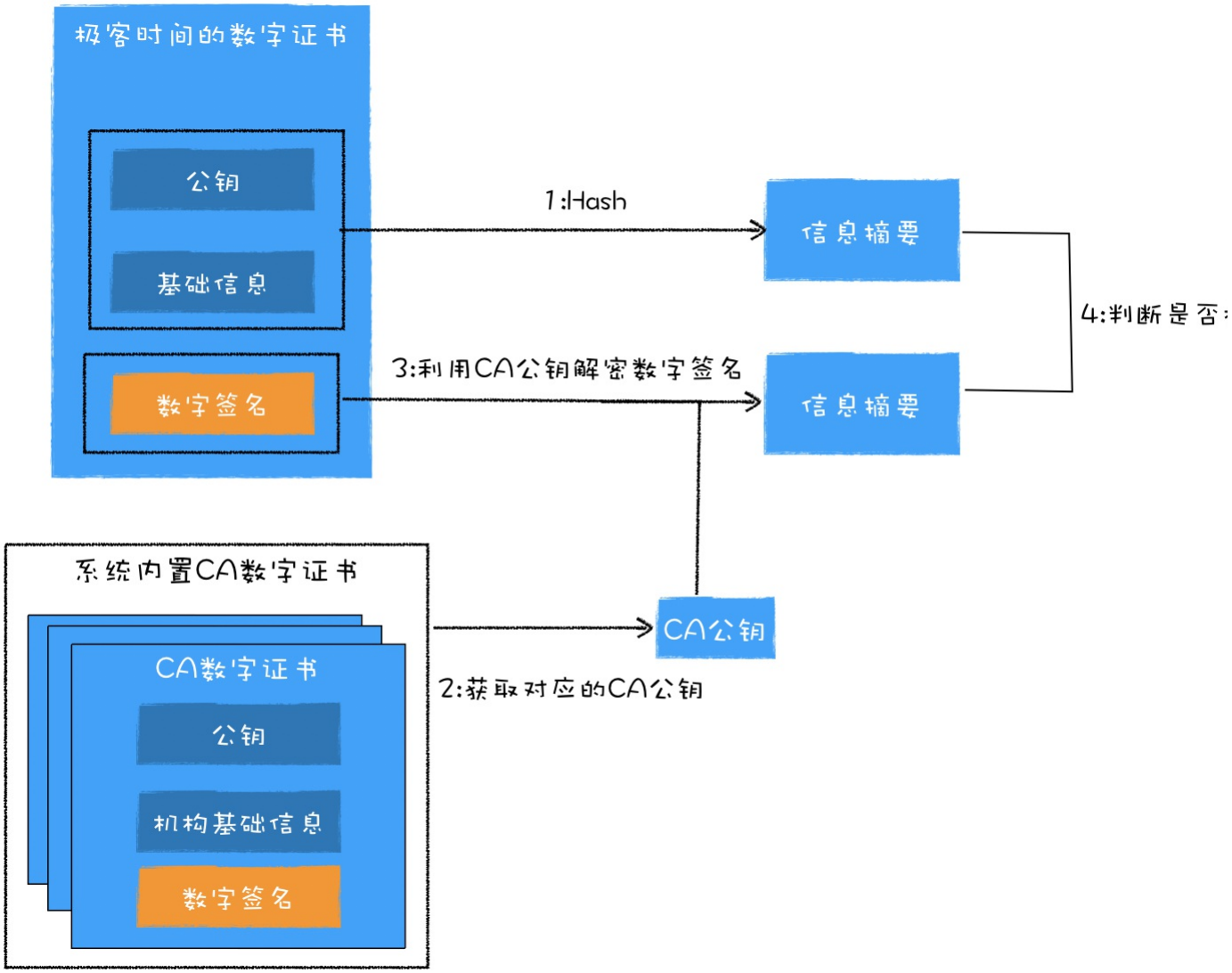
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

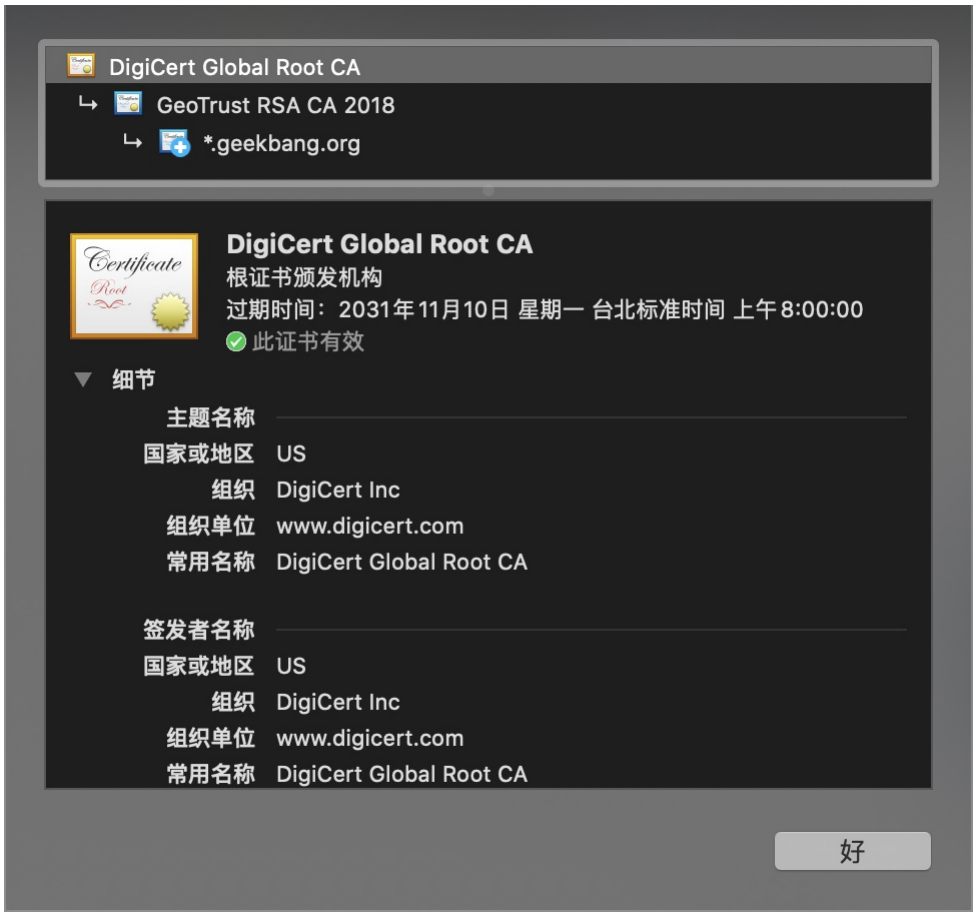
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

你好，我是李兵。

在《36丨HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

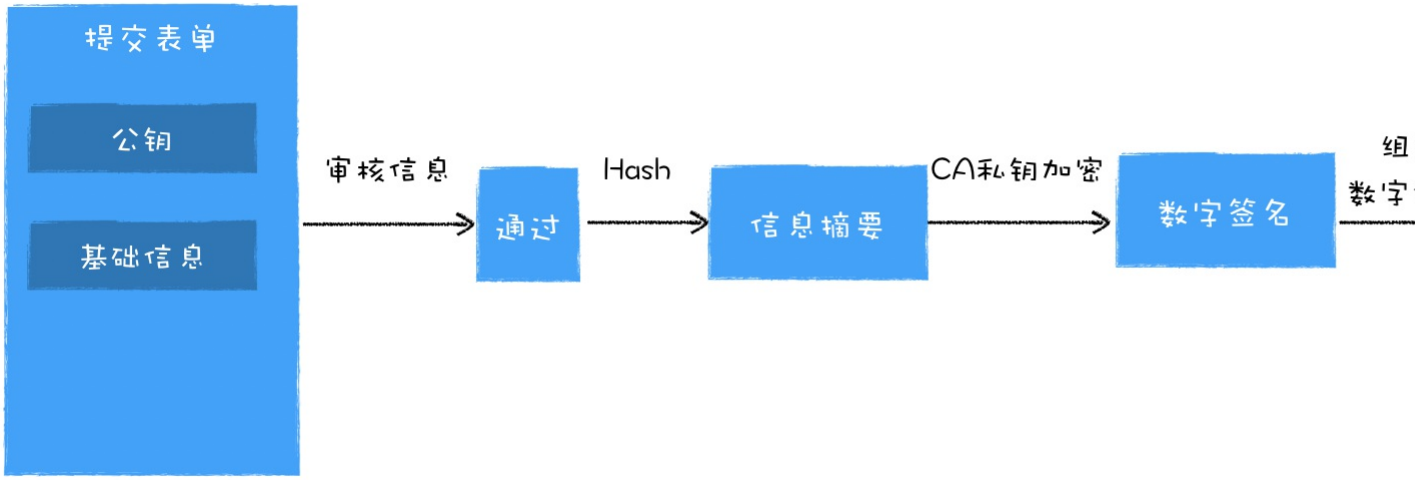
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

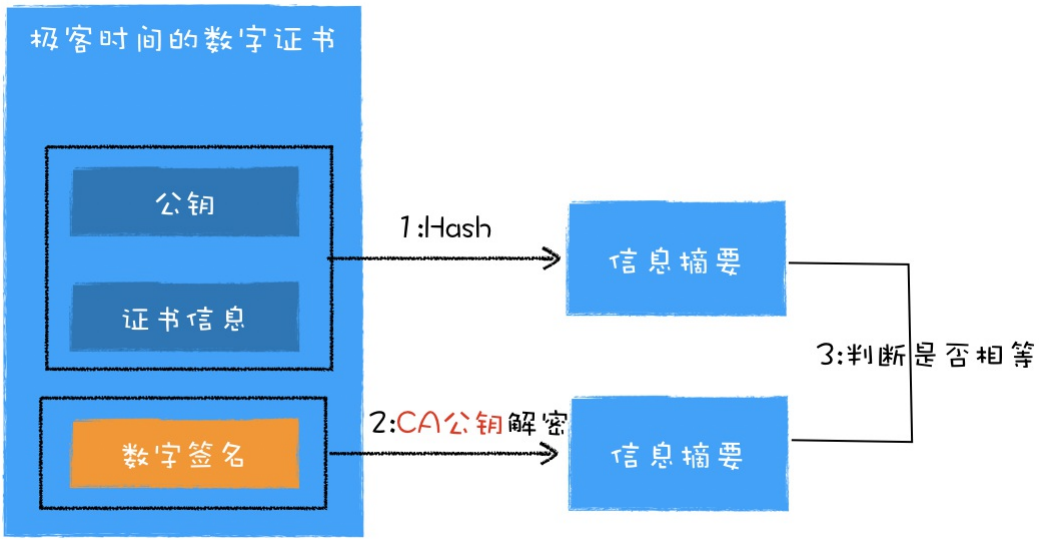
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

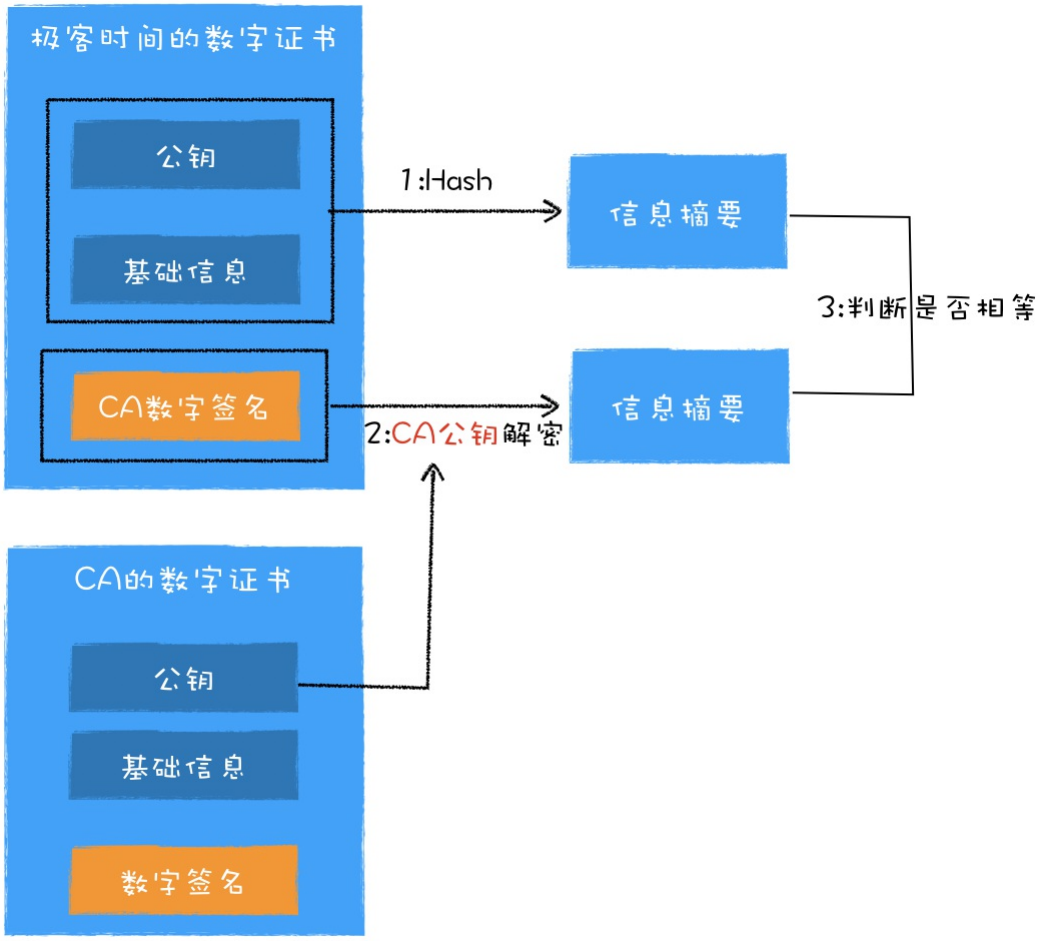
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



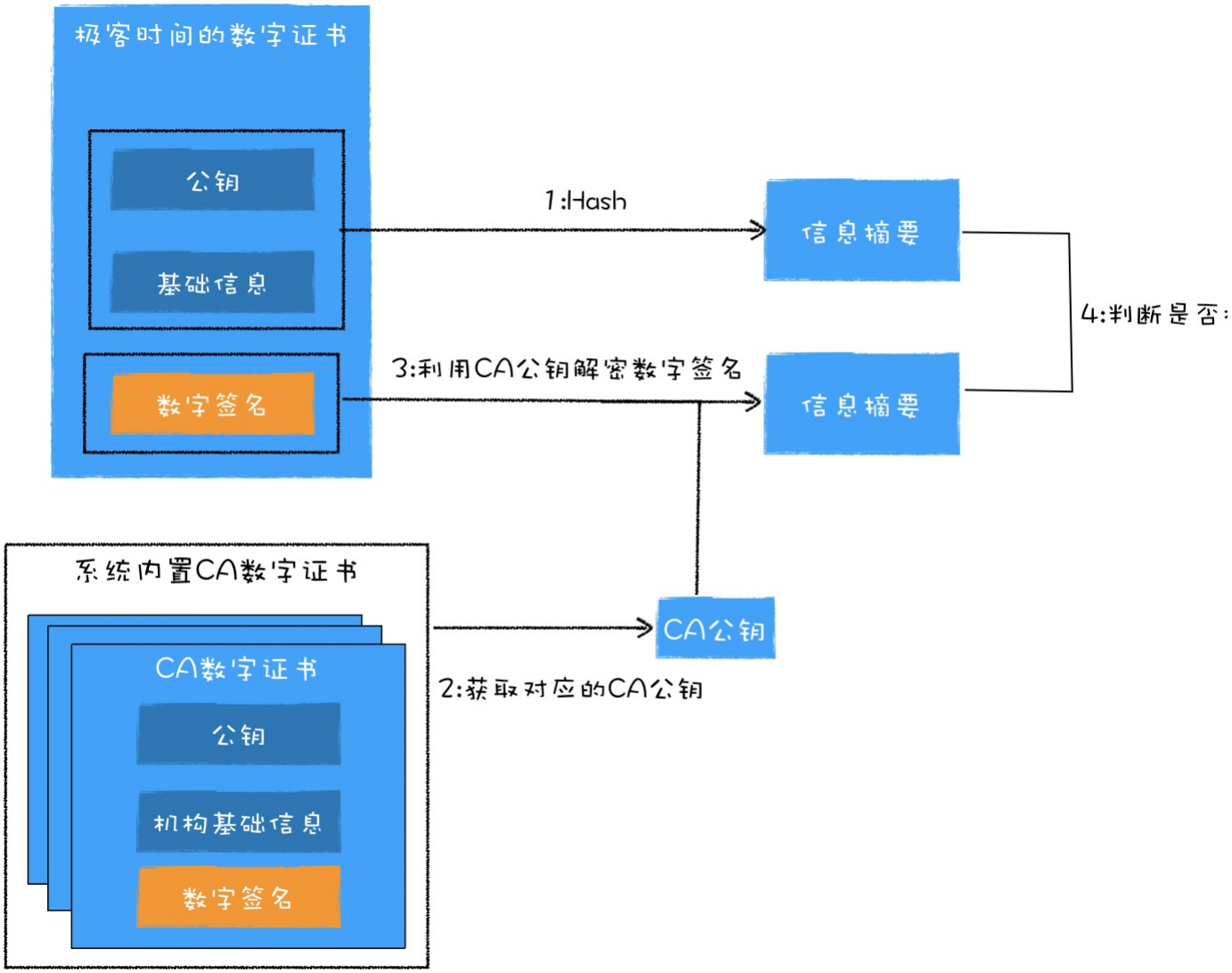
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

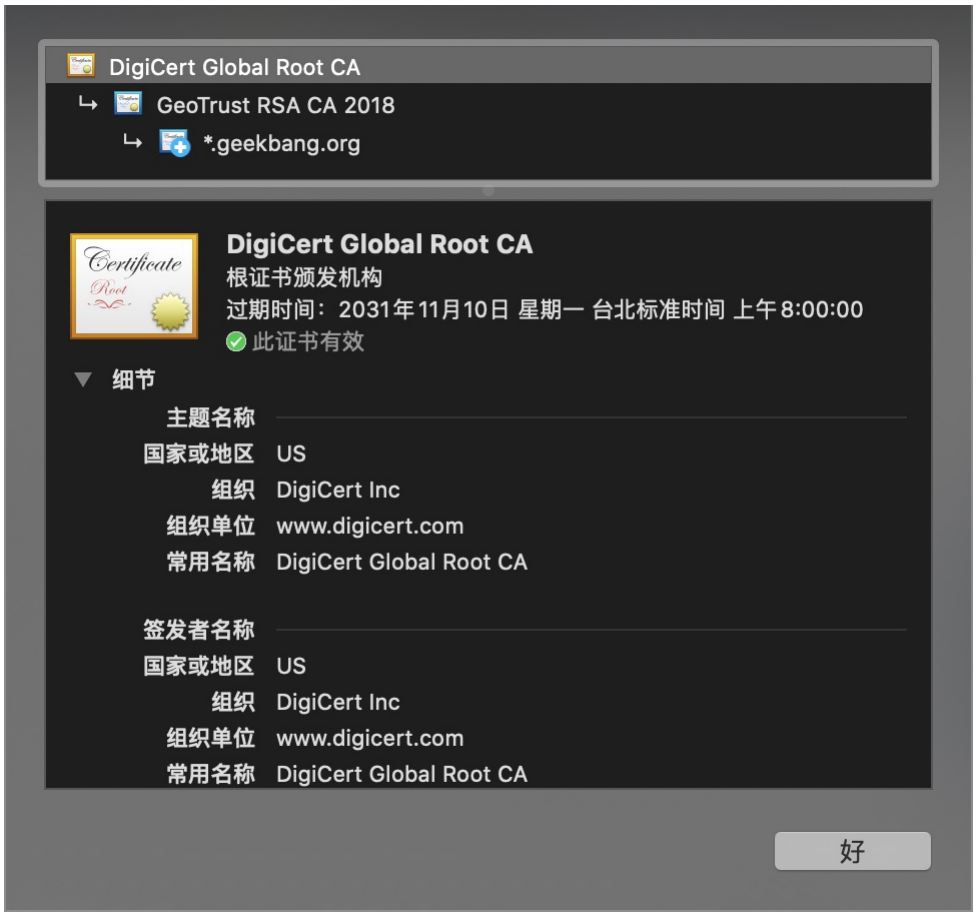
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

你好，我是李兵。

在《36丨HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

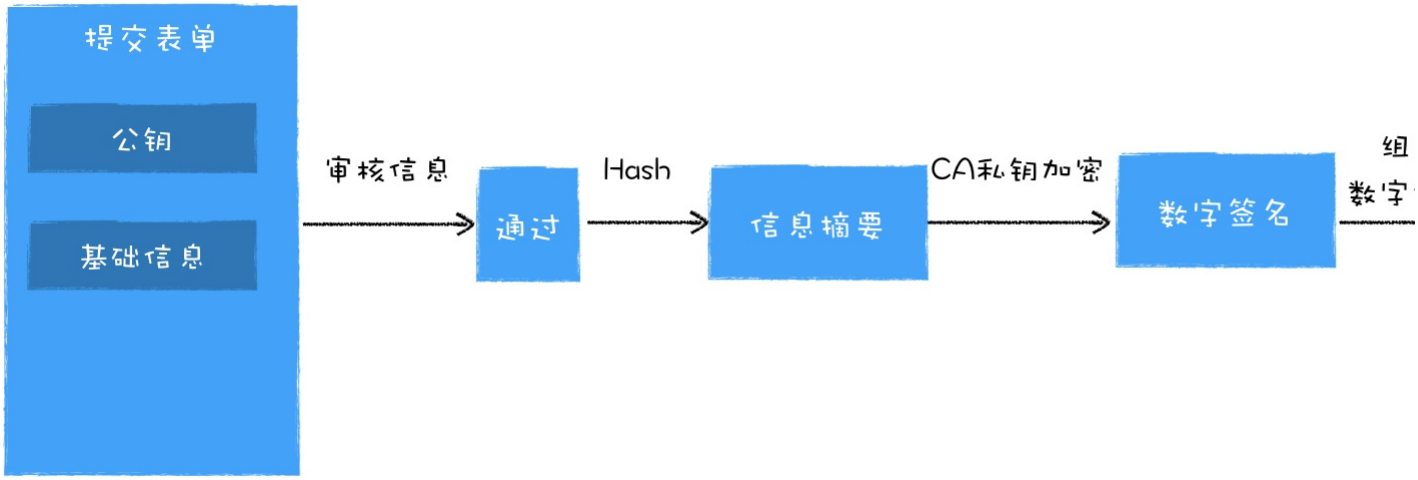
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

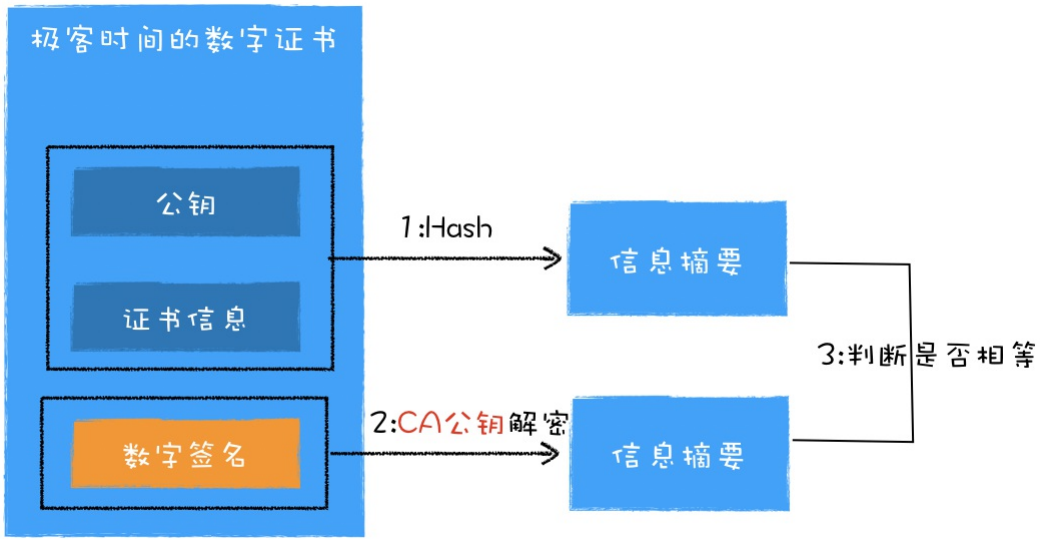
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

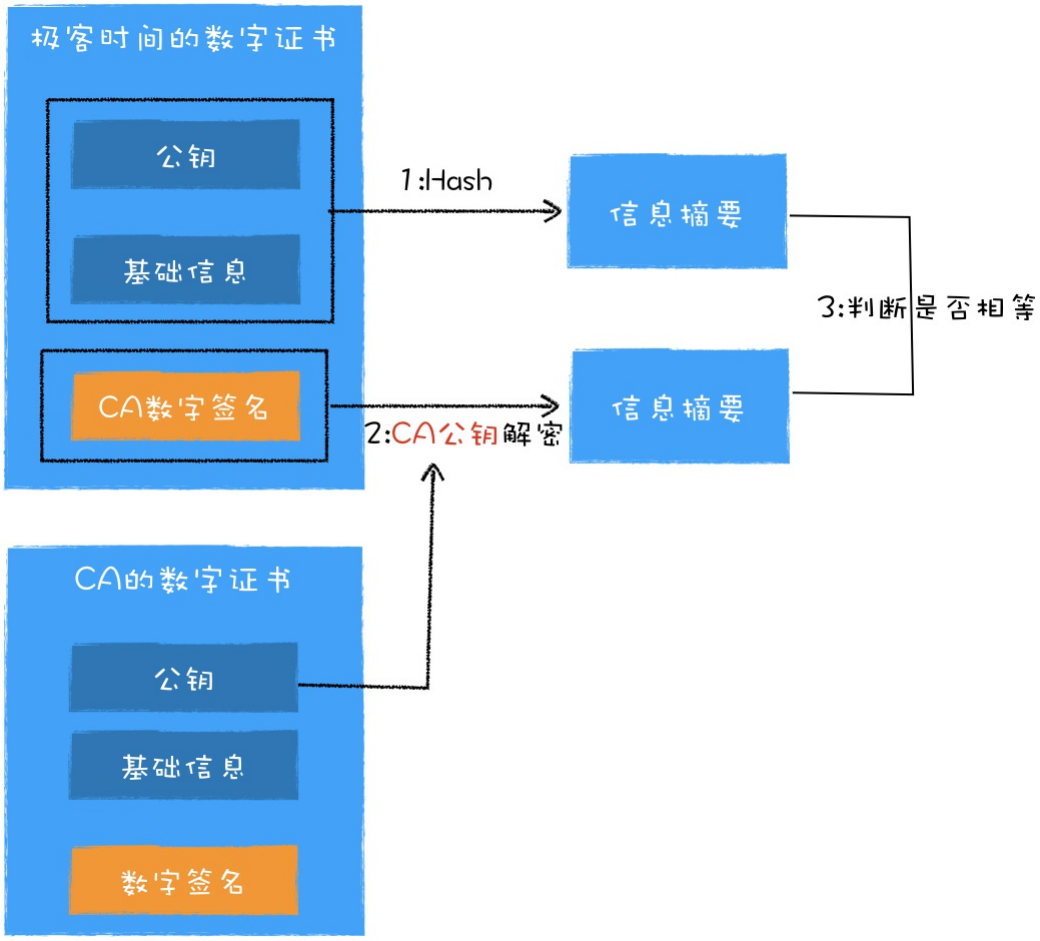
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



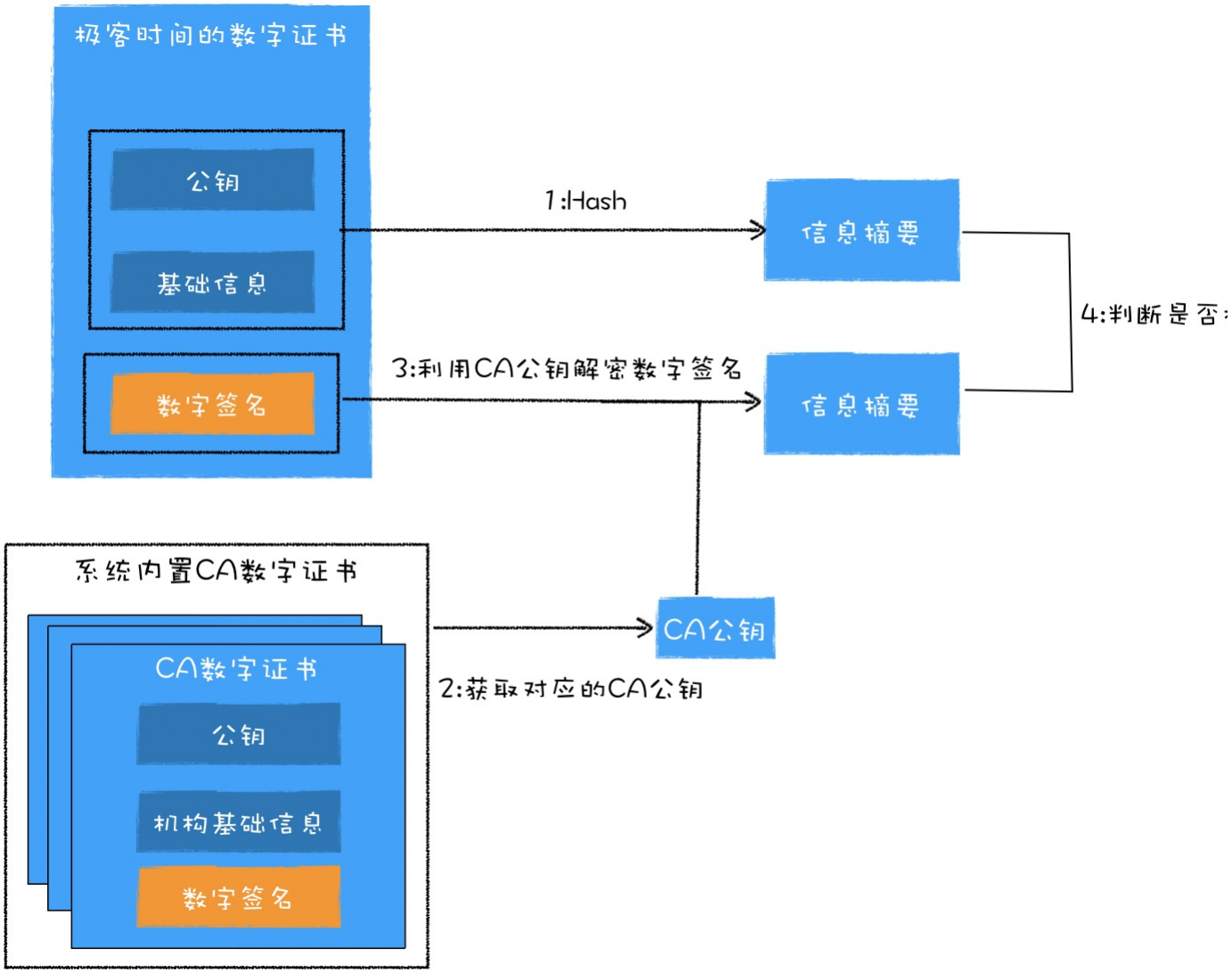
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

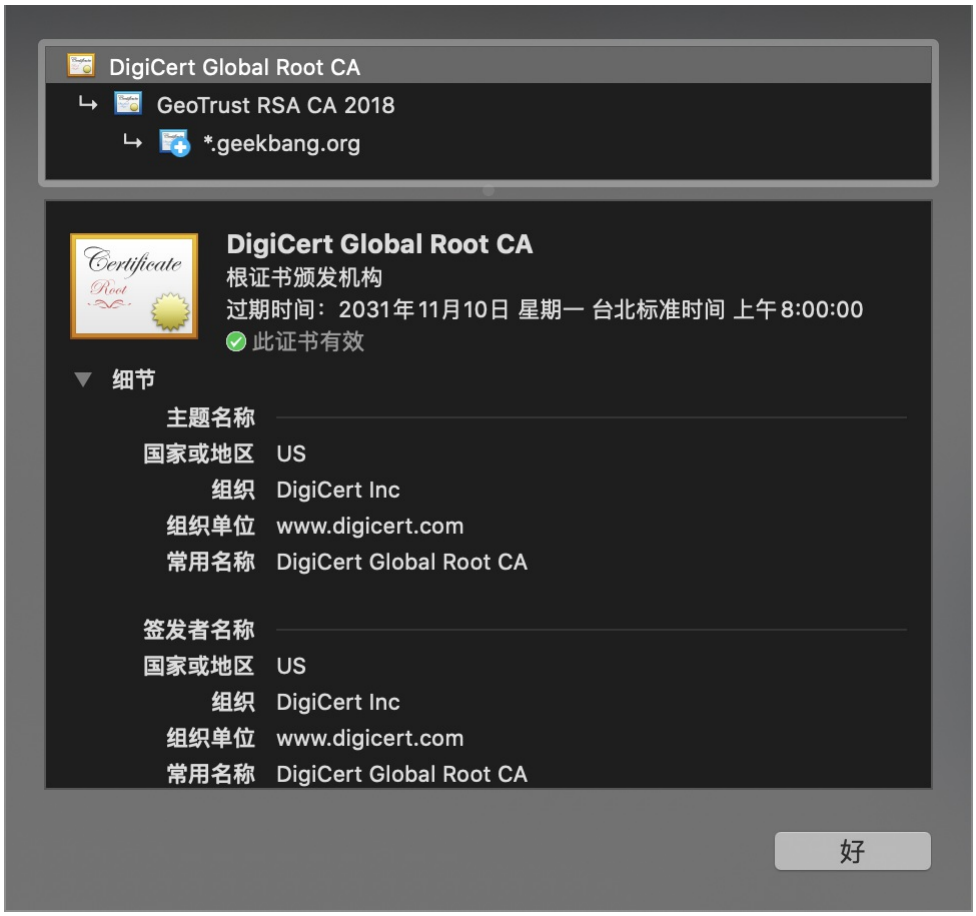
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。

你好，我是李兵。

在《36丨HTTPS：让数据传输更安全》这篇文章中，我们聊了下面几个问题：

- HTTPS使用了对称和非对称的混合加密方式，这解决了数据传输安全的问题；
- HTTPS引入了中间机构CA，CA通过给服务器颁发数字证书，解决了浏览器对服务器的信任问题；
- 服务器向CA机构申请证书的流程；
- 浏览器验证服务器数字证书的流程。

不过由于篇幅限制，关于**“浏览器如何验证数字证书”**的这个问题我们并没有展开介绍。那么今天我们就继续聊一聊这个问题。了解了这个问题，可以方便我们把完整的HTTPS流程给串起来，无论对于我们理解HTTPS的底层技术还是理解业务都是非常有帮助的。

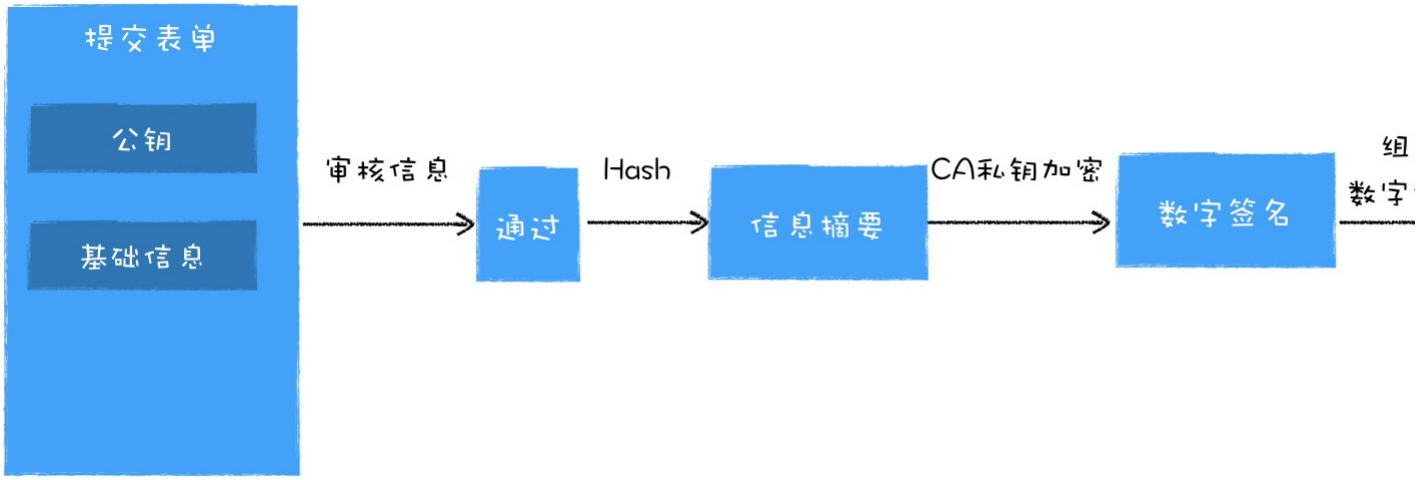
因为本文是第36讲的延伸，所以在分析之前，我们还是有必要回顾下**数字证书申请流程**和**浏览器验证证书的流程**，同时你最好也能回顾下第36讲。

数字证书申请流程

我们先来回顾下数字证书的申请流程，比如极客时间向一个CA机构申请数字证书，流程是什么样的呢？

首先极客时间填写了一张含有**自己身份信息**的表单，身份信息包括了自己公钥、站点资料、公司资料等信息，然后将其提交给了CA机构；CA机构会审核表单中内容的真实性；审核通过后，CA机构会拿出自己的私钥，对表单的内容进行一连串操作，包括了对明文资料进行Hash计算得出信息摘要， 利用CA的私钥加密信息摘要得出数字签名，最后将数字签名也写在表单上，并将其返还给极客时间，这样就完成了一次数字证书的申请操作。

大致流程你也可以参考下图：



数字证书申请过程

浏览器验证证书的流程

现在极客时间的官网有了CA机构签发的数字证书，那么接下来就可以将数字证书应用在HTTPS中了。

我们知道，在浏览器和服务器建立HTTPS链接的过程中，浏览器首先会向服务器请求数字证书，之后浏览器要做的第一件事就是验证数字证书。那么，这里所说的“验证”，它到底是在验证什么呢？

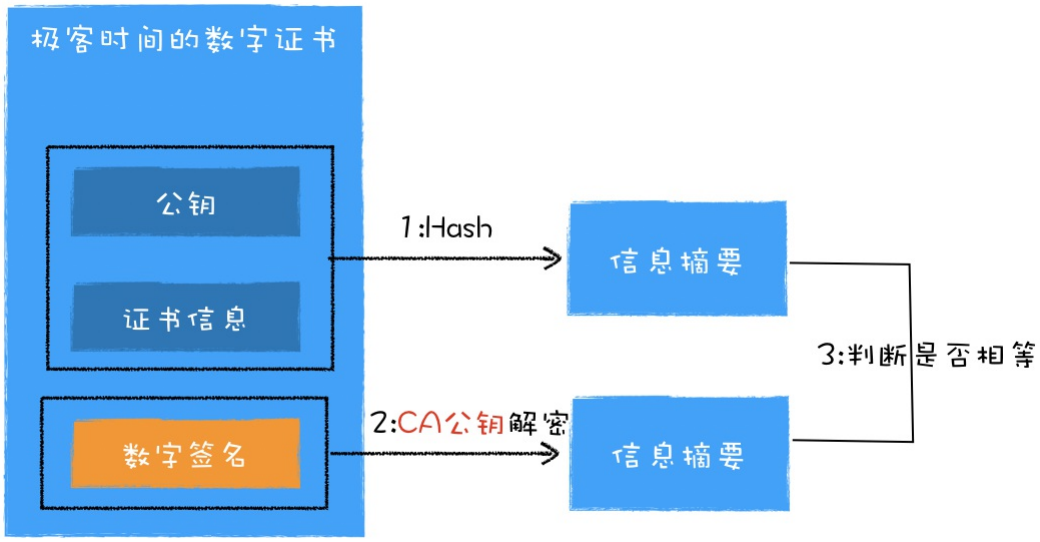
具体地讲，浏览器需要验证**证书的有效期、证书是否被CA吊销、证书是否是合法的CA机构颁发的。**

数字证书和身份证一样也是有时间期限的，所以**第一部分就是验证证书的有效期**，这部分比较简单，因为证书里面就含有证书的有效期，所以浏览器只需要判断当前时间是否在证书的有效期范围内即可。

有时候有些数字证书被CA吊销了，吊销之后的证书是无法使用的，所以**第二部分就是验证数字证书是否被吊销了**。通常有两种方式，一种是下载吊销证书列表-CRL (Certificate Revocation Lists)，第二种是在线验证方式-OCSP (Online Certificate Status Protocol)，它们各有优缺点，在这里我就不展开介绍了。

最后，还要**验证极客时间的数字证书是否是CA机构颁发的，**验证的流程非常简单：

- 首先，浏览器利用证书的原始信息计算出信息摘要；
- 然后，利用**CA的公钥**来解密数字证书中的**数字签名**，解密出来的数据也是信息摘要；
- 最后，判断这两个信息摘要是否相等就可以了。



通过这种方式就验证了数字证书是否是由CA机构所签发的，不过这种方式又带来了一个新的疑问：**浏览器是怎么获取到CA公钥的**？

浏览器是怎么获取到CA公钥的？

通常，当你部署HTTP服务器的时候，除了部署当前的数字证书之外，还需要部署CA机构的数字证书，CA机构的数字证书包括了CA的公钥，以及CA机构的一些基础信息。

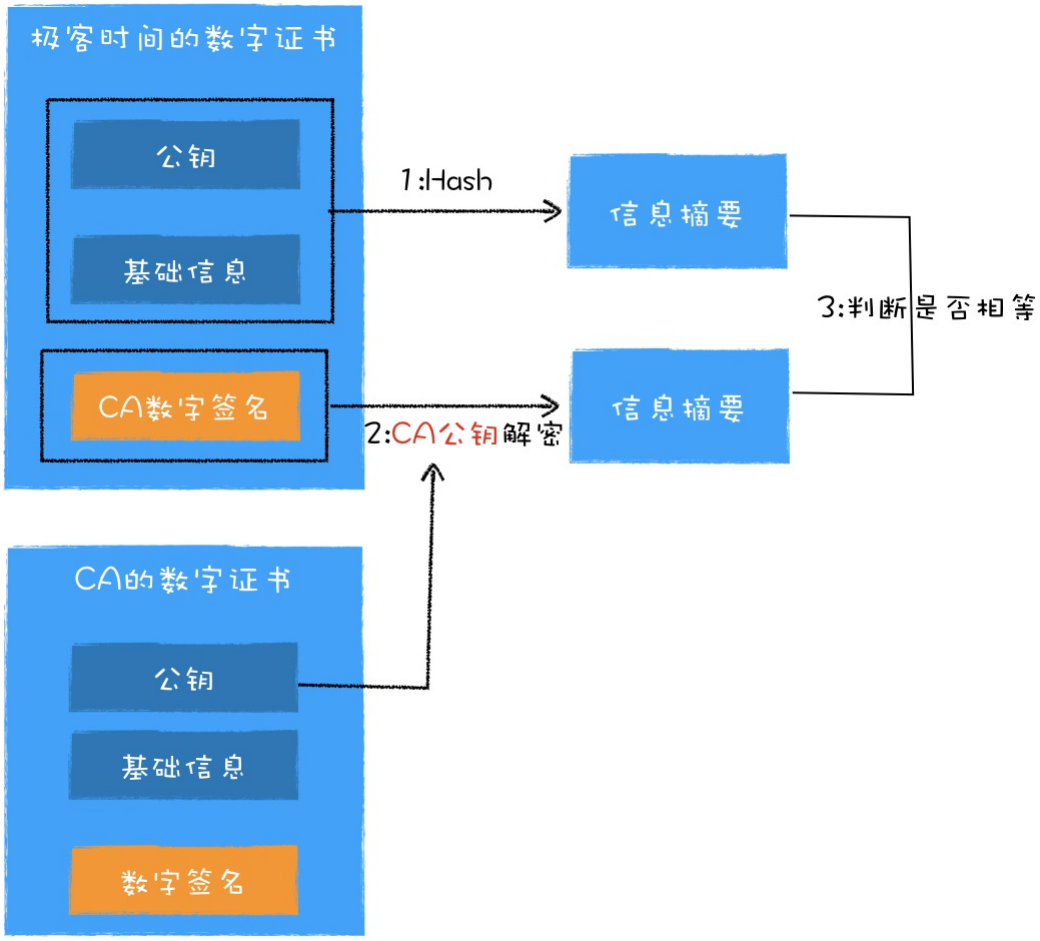
因此，极客时间服务器就有了两个数字证书:

- 给极客时间域名的数字证书；
- 给极客时间签名的CA机构的数字证书。

然后在建立HTTPS链接时，服务器会将这两个证书一同发送给浏览器，于是浏览器就可以获取到CA的公钥了。

如果有些服务器没有部署CA的数字证书，那么浏览器还可以通过网络去下载CA证书，不过这种方式多了一次证书下载操作，会拖慢首次打开页面的请求速度，一般不推荐使用。

现在浏览器端就有了极客时间的证书和CA的证书，完整的验证流程就如下图所示：



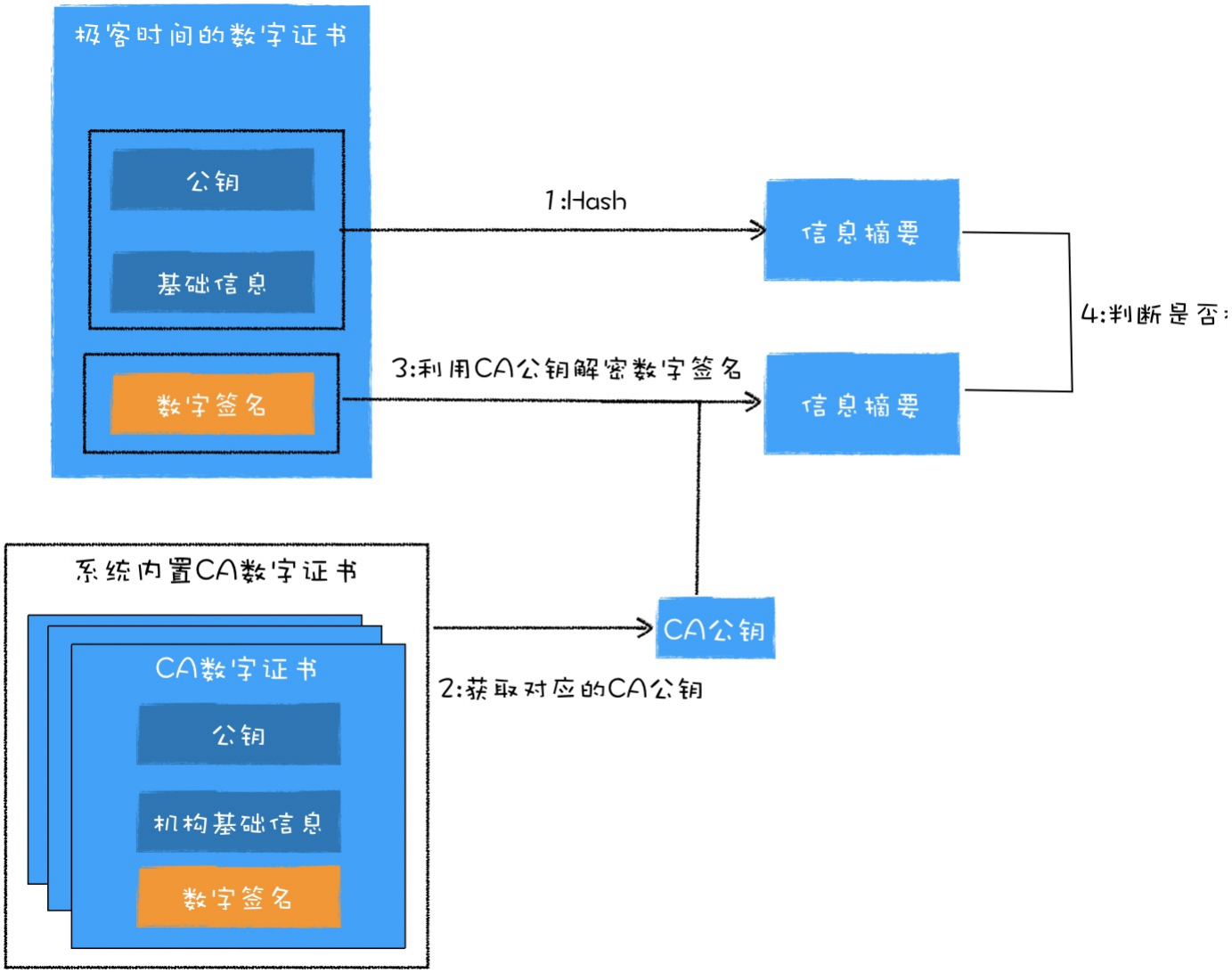
CA证书

我们有了CA的数字证书，也就可以获取得CA的公钥来验证极客时间数字证书的可靠性了。

解决了获取CA公钥的问题，新的问题又来了，如果这个证书是一个恶意的CA机构颁发的怎么办？所以我们还需要**浏览器证明这个CA机构是个合法的机构**。

证明CA机构的合法性

这里并没有一个非常好的方法来证明CA的合法性，妥协的方案是，直接在操作系统中内置这些CA机构的数字证书，如下图所示：



操作系统内部内置CA数字证书

我们将所有CA机构的数字证书都内置在操作系统中，这样当需要使用某CA机构的公钥时，我们只需要依据CA机构名称，就能查询到对应的数字证书了，然后再从数字证书中取出公钥。

可以看到，这里有一个假设条件，浏览器默认信任操作系统内置的证书为合法证书，虽然这种方式不完美，但是却是最实用的一个。

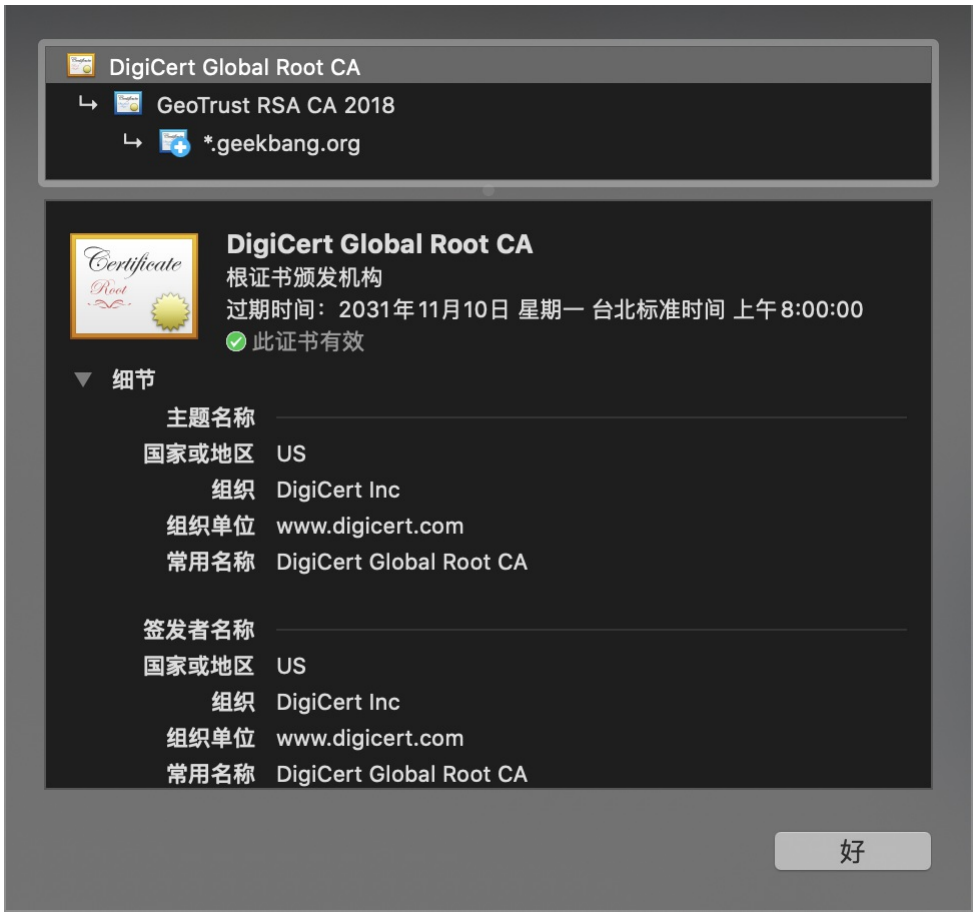
不过这种方式依然存在问题，因为在实际情况下，**CA机构众多，因此操作系统不可能将每家CA的数字证书都内置进操作系统**。

数字证书链

于是人们又想出来一个折中的方案，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

因此，每个根CA机构都维护了一个树状结构，一个根CA下面包含多个中间CA，而中间CA又可以包含多个中间CA。这样就形成了一个证书链，你可以沿着证书链从用户证书追溯到根证书。

比如你可以在Chrome上打开极客时间的官网，然后点击地址栏前面的那把小锁，你就可以看到*.geekbang.org的证书是由中间CA GeoTrust RSA CA2018颁发的，而中间CA GeoTrust RSA CA2018又是由根CA DigiCert Global Root CA颁发的，所以这个证书链就是：*.geekbang.org—>GeoTrust RSA CA2018–>DigiCert Global Root CA。你可以参看下图：



数字证书链

因此浏览器验证极客时间的证书时，会先验证*.geekbang.org的证书，如果合法，再验证中间CA的证书，如果中间CA也是合法的，那么浏览器会继续验证这个中间CA的根证书。

到了这里，依然存在一个问题，那就是**浏览器怎么证明根证书是合法的？**

如何验证根证书的合法性

其实浏览器的判断策略很简单，它只是简单地判断这个根证书在不在操作系统里面，如果在，那么浏览器就认为这个根证书是合法的，如果不在，那么就是非法的。

如果某个机构想要成为根CA，并让它的根证书内置到操作系统中，那么这个机构首先要通过WebTrust国际安全审计认证。

什么是WebTrust认证？

WebTrust是由两大著名注册会计师协会AICPA（美国注册会计师协会）和CICA（加拿大注册会计师协会）共同制定的安全审计标准，主要对互联网服务商的系统及业务运作逻辑安全性、保密性等共计七项内容进行近乎严苛的审查和鉴证。 只有通过WebTrust国际安全审计认证，根证书才能预装到主流的操作系统，并成为一个可信的认证机构。

目前通过WebTrust认证的根CA有 Comodo、geotrust、rapidssl、symantec、thawte、digicert等。也就是说，这些根CA机构的根证书都内置在个大操作系统中，只要能从数字证书链往上追溯到这几个根证书，浏览器就会认为使用者的证书是合法的。

总结

好了，今天的内容就介绍到这里，下面我们总结下本文的主要内容：

我们先回顾了数字证书的申请流程，接着我们重点介绍了浏览器是如何验证数字证书的。

首先浏览器需要CA的数字证书才能验证极客时间的数字证书，接下来我们需要验证CA证书的合法性，最简单的方法是将CA证书内置在操作系统中。

不过CA机构非常多，内置每家的证书到操作系统中是不现实的，于是我们采用了一个折中的策略，将颁发证书的机构划分为两种类型，**根CA(Root CAs)和中间CA(Intermediates CAs)**，通常申请者都是向中间CA去申请证书的，而根CA作用就是给中间CA做认证，一个根CA会认证很多中间的CA，而这些中间CA又可以去认证其他的中间CA。

于是又引出了数字证书链，浏览器先利用中间CA的数字证书来验证用户证书，再利用根证书来验证中间CA证书的合法性，最后，浏览器会默认相信内置在系统中的根证书。不过要想在操作系统内部内置根证书却并不容易，这需要通过WebTrust认证，这个认证审核非常严格。

通过分析这个流程可以发现，浏览器默认信任操作系统内置的根证书，这也会带来一个问题，如果黑客入侵了你的电脑，那么黑客就有可能往你系统中添加恶意根数字证书，那么当你访问黑客站点的时候，浏览器甚至有可能会提示该站点是安全的。

因此，HTTPS并非是绝对安全的，采用HTTPS只是加固了城墙的厚度，但是城墙依然有可能被突破。

课后思考

今天留给你的任务是复述下浏览器是怎么验证数字证书的，如果中间卡住了，欢迎在留言区提问交流。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。