Certificate Pinning in iOS

Karol Piątek
Jan 17, 2020 | 12 min read
Mobile (/codestories/topic/mobile) iOS (/codestories/topic/ios) Swift (/codestories/topic/swift) Security (/codestories/topic/security)

Introduction

Certificate pinning is one of the basic security mechanisms of network communication. Every developer should be aware of it. The OWASP security organization includes it in their "General Best Practices" and "iOS Specific Best Practices (https://www.owasp.org/index.php/Mobile_Top_10_2016-M3-Insecure_Communication)". This shows how important this topic is.

This article discusses certificate pinning and related topics.

You will get familiar with things like:

- TLS/SSL and its weaknesses
- Certificate pinning
- What is recommended and how to implement it on iOS

Theory

What is TLS? (Transport Layer Security)

TLS is an extended version of the SSL protocol. It is a cryptographic protocol used to encrypt network traffic. Your browser uses an implementation of that protocol - HTTPS. It is usually displayed next to the website address as a padlock icon.

How TLS works?

Formal definition

TLS uses asymmetric cryptography to provide secure data transportation. Asymmetric cryptography uses two keys, a public key and a private key. The public key is used to encrypt data and the private key decrypts previously encrypted data. When you make a connection with a server you exchange public keys with it. You receive the public key to encrypt data before sending it. The server receives your public key so you can decrypt the data received from it with your private key. The keys are uniquely generated for each connection and are based on a shared secret negotiated at the beginning of the session, also known as a TLS handshake.

Example by analogy

TLS works like a mailbox. Everyone has access to the mailbox and can put a letter inside it, but you need a key to open the mailbox and read letters. Only the person who created the mailbox has the key. TLS is considered undecryptable in reasonable time. This means that the best locksmith would have to work on it for a long time to open it without having a key.

You can also check out this video (https://www.youtube.com/watch?v=E5FEqGYLL0o) visualising asymmetric encryption.

Some Dates

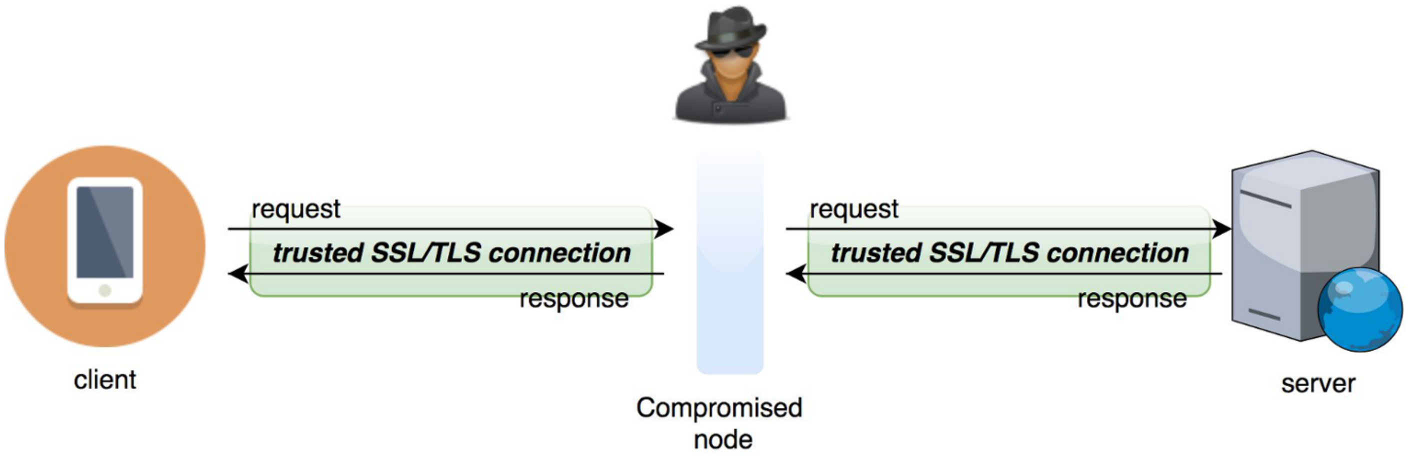
- SSL protocol is deprecated since 2015.
- TLS 1.0/1.1 will be deprecated in 2020.
- There are no plans to deprecate TLS 1.2/1.3 yet.

What is a certificate?

A certificate is a data file containing a public key and other information like expiration date, organization name, and others. It allows us to encrypt a connection using the public key.

So what is the problem with using only the TLS protocol for networking?

Imagine you are using a public Wi-Fi network on the train. That network might be created by someone who wants to read the data sent by you. Since that person is providing your internet connection, they can create their own certificate and tell you to encrypt your data with it. Because this person created the certificate, they can read all the data inside it and then send it further to the right receiver. You may not see that you're using the wrong certificate. This technique is called a man-in-the-middle attack. This illustration shows how it works.



How to prevent using invalid certificates?

This is where certificate pinning and public key pinning come in. These approaches consist of defining a list of valid certificates/public keys. Every time the app sends data, it checks if that certificate is on the list of trusted certificates/public keys. This list of certificates/public keys may be hard-coded or defined in a file. Both solutions prevent man-in-the-middle attacks and should pass a penetration test. If you want to be even more secure, you may also encrypt the certificate to make it harder to get.

More formal definition:

"Server/Host is associated with a certificate or public key. You configure the app to reject all certificates other than the predefined certificates or public keys. Whenever the app connects to a server, it compares the server certificate with the pinned certificate(s) or public key(s). If and only if they match, the app trusts the server and establishes the connection."

Certificate pinning vs public key pinning

Public key pinning is a simplified implementation of certificate pinning. Unlike certificate pinning, It only validates the public key instead of the whole certificate.

The public key doesn't have an expiration date. So if we have implemented public key pinning, we can regenerate your certificate with the same public key. This allows us to regenerate the certificate without uploading a new version of the app. But, since it doesn't change, it may violate key rotation policies.

Then there is certificate pinning. It validates the whole certificate, so this implementation is more secure, since even a small change in the certificate will cause an issue. It has an expiration date, so every time you create a new certificate you have to upload a new version of the app.

Usually, the heavier your security implementation, the less flexible and harder to maintain it gets. You should always look for a solution that fits your app.

- If you really care about security and you can do it at the expense of flexibility, you should choose hardcoded certificate pinning. This solution should be used for example in banking applications - it is the safest of those presented here.
- If your certificates change very often and security isn't crucial in your project, you should probably use public key pinning. It will give you more flexibility and you will not block users with older versions of your application.
- If you want to support older versions of the app, but still implement certificate pinning, you can create a procedure to download the new certificate after expiration. It is not as safe as a hardcoded one, but it gives you more flexibility. After the certificate expires, you will download the new one. While you are downloading the new certificate, there is a chance that someone is listening to your communication.
- If your application does not need additional protection, e.g. it uses a public API to display the weather, you do not have to implement certificate pinning.

Is that all? Am I safe now?

Well, no. There are still techniques to fool certificate pinning, but using it makes it extremely hard to decrypt network traffic. You can read about some bypasses here: Certificate pinning bypass (https://blog.netspi.com/four-ways-to-bypass-ios-ssl-verification-and-certificate-pinning/).

What is recommended for iOS?

AlamoFire (https://github.com/Alamofire/Alamofire) is a good choice when you are already using that or you are looking for a network library. If networking in your app uses NSURLSession and you only need certificate pinning, TrustKit (https://github.com/datatheorem/TrustKit) is a good option.

You shouldn't implement pinning by yourself, as implementation mistakes are extremely likely and usually lead to severe vulnerabilities.

OWASP's stance on certificate pinning

OWASP (Open Web Application Security Project) is an organization that provides unbiased and practical, cost-effective information about computer and Internet applications. This organization is well known. A lot of big companies use "OWASP top 10 application security vulnerabilities" in their pentesting plans.

You can find that certificate pinning is listed among the "General Best Practices" and "iOS Specific Best Practices (https://www.owasp.org/index.php/Mobile_Top_10_2016-M3-Insecure_Communication)" on the OWASP website.

"You should pin anytime you want to be relatively certain of the remote host's identity or when operating in a hostile environment. Since one or both are almost always true, you should probably pin all the time." - OWASP pinning cheat sheet (https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Pinning_Cheat_Sheet.md)

Using certificate pinning in practice

How to get an example certificate.der?

Run this command in your terminal, you should get Google's certificate with the .der extension:

```
openssl s_client -connect google.com:443 </dev/null | openssl x509 -outform DER -out google.com.der
```

Alamofire certificate pinning

The Alamofire library has a built-in class to handle certificate pinning: ServerTrustManager.

Let's see the initialize method of this class:

```
init(allHostsMustBeEvaluated: Bool , evaluators: [String: ServerTrustEvaluating])
```

Evaluators - a dictionary with the website address as a key and the list of defined certificates wrapped in the ServerTrustEvaluating class as a value of it.

allHostsMustBeEvaluated: Bool -

If true: Alamofire will only allow communication with hosts defined in the evaluators and matching the defined certificates.

If false: Alamofire will check certificates only for hosts defined in the evaluators dictionary. Communication with other hosts will not use certificate pinning

Let's create a ServerTrustManager for your project!

Add your certificate to your project and add an extension class to read that certificate.

```
struct Certificates {

    static let certificate: SecCertificate = Certificates.certificate(filename: "certificateFileName")

    private static func certificate(filename: String) -> SecCertificate {

        let filePath = Bundle.main.path(forResource: filename, ofType: "fileExtension")!
        let data = try! Data(contentsOf: URL(fileURLWithPath: filePath))
        let certificate = SecCertificateCreateWithData(nil, data as CFData)!

        return certificate
  }
}
```

Now let's set up the networking class, which will pin that certificate.

```
import Alamofire

class AlamofireNetworking {

    private let certificates = [
      "www.yourwebsite.com":
        PinnedCertificatesTrustEvaluator(certificates: [Certificates.certificate],
                                         acceptSelfSignedCertificates: false,
                                         performDefaultValidation: true,
                                         validateHost: true)
    ]

    private let session: Session

    init(allHostsMustBeEvaluated: Bool) {

        let serverTrustPolicy = ServerTrustManager(
            allHostsMustBeEvaluated: allHostsMustBeEvaluated,
            evaluators: certificates
        )

        session = Session(serverTrustManager: serverTrustPolicy)
    }

    func request(_ convertible: URLRequestConvertible) -> DataRequest {
      return session.request(convertible)
    }
}
```

The implementation is really easy and readable here. I you use the request method from that class, certificate pinning will work. You can see the usage of this class in the unit tests of the project (https://github.com/karolpiateknet/CertificatePinning). You can also read the documentation (https://github.com/Alamofire/Alamofire/blob/master/Documentation/AdvancedUsage.md) for more information.

TrustKit certificate pinning

First, install TrustKit in your project. You can do this using cocoaPods or carthage. Use the instruction (https://datatheorem.github.io/TrustKit/getting-started.html#deploying-trustkit)s to do this.

In order to implement certificate pinning with TrustKit, we have to define the configuration for this tool. To do so, we need to extract publicKeyHashes from our domain. You can easily get them using script (https://github.com/datatheorem/TrustKit/blob/master/get_pin_from_certificate.py) provided by TrustKit. Just follow the instruction (https://datatheorem.github.io/TrustKit/getting-started.html#generating-ssl-pins)s in their documentation.

Here is a complete class with an URLSession which uses certificate pinning.

```
final class TrustKitCertificatePinning: NSObject, URLSessionDelegate {

    /// URLSession with configured certificate pinning
    lazy var session: URLSession = {
        URLSession(configuration: URLSessionConfiguration.ephemeral,
                   delegate: self,
                   delegateQueue: OperationQueue.main)
    }()

    private let trustKitConfig = [
        kTSKPinnedDomains: [
            "www.yourwebsite.com": [
                kTSKEnforcePinning: true,
                kTSKIncludeSubdomains: true,
                kTSKExpirationDate: "2020-10-09",
                kTSKPublicKeyHashes: [
                    "GesrhCSBz+OCxCt624nic/qqLXAPGUGGf5vwB5jBheU=",
                    "+7YVLndnzqU0VtEREXo00bJlgdmQ9T9qy2IWVVWNpcE=",
                ],
            ]
        ]
    ] as [String : Any]

    override init() {
        TrustKit.initSharedInstance(withConfiguration: trustKitConfig)
        super.init()
    }

    // MARK: TrustKit Pinning Reference

    func urlSession(_ session: URLSession,
                    didReceive challenge: URLAuthenticationChallenge,
                    completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

        if TrustKit.sharedInstance().pinningValidator.handle(challenge, completionHandler: completionHandler) == false {
            // TrustKit did not handle this challenge: perhaps it was not for server trust
            // or the domain was not pinned. Fall back to the default behavior
            completionHandler(.performDefaultHandling, nil)
        }
    }
}
```

As you can see, there is not much to talk about here, but let's dig into that code.

This part of code defines configuration for TrustKit. You can define the domains that you want to check.

```
private let trustKitConfig = [
        kTSKPinnedDomains: [
            "www.yourwebsite.com": [
                kTSKEnforcePinning: true,
                kTSKIncludeSubdomains: true,
                kTSKExpirationDate: "2020-10-09",
                kTSKPublicKeyHashes: [
                    "GesrhCSBz+OCxCt624nic/qqLXAPGUGGf5vwB5jBheU=",
                    "+7YVLndnzqU0VtEREXo00bJlgdmQ9T9qy2IWVVWNpcE=",
                ],
            ]
        ]
    ] as [String : Any]
```

You can read more about configuration in the documentation (https://github.com/datatheorem/TrustKit).

To get TrustKit working, you have to initialize a TrustKit shared instance with this configuration.

```
TrustKit.initSharedInstance(withConfiguration: trustKitConfig)
```

Then you can use that sharedInstance in the urlSession delegate methods to validate your certificates. This is the default implementation from TrustKit's documentation.

```
func urlSession(_ session: URLSession,
                    didReceive challenge: URLAuthenticationChallenge,
                    completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

    if TrustKit.sharedInstance().pinningValidator.handle(challenge, completionHandler: completionHandler) == false {
        // TrustKit did not handle this challenge: perhaps it was not for server trust
        // or the domain was not pinned. Fall back to the default behavior
        completionHandler(.performDefaultHandling, nil)
    }
}
```

The last thing you need to do is assign the created Delegate to your session.

```

final class TrustKitCertificatePinning: NSObject, URLSessionDelegate {

    /// URLSession with configured certificate pinning
    lazy var session: URLSession = {
        URLSession(configuration: URLSessionConfiguration.ephemeral,
                   delegate: self,
                   delegateQueue: OperationQueue.main)
     }()
```

Use your URLSession with certificate pinning :)

Complete CertificatePinning project with unit tests:

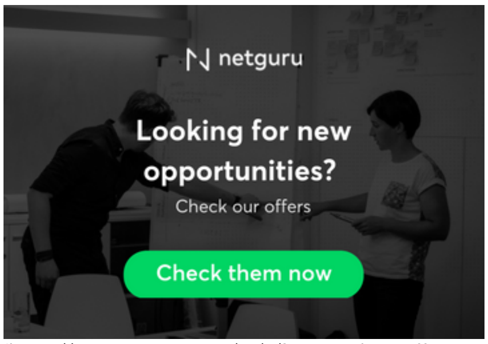
https://github.com/karolpiateknet/CertificatePinning (https://github.com/karolpiateknet/CertificatePinning)

Summary

Thank you for your time. I hope you enjoyed reading my article and learned a lot of useful information.

Karol Piątek

(https://www.netguru.com/cs/c/?cta_guid=b3a2d25e-4b17-440f-809c-0a803790c0b7&signature=AAH58kF6nyyzk-o9yaDOVWxafmhL9rOcNw&pageId=23961616481&placement_guid=f869f20d-aae9-4efc-8189-5464e3bf47a1&click=0306e888-6829-4c15-b935-ede2273ce7c8&hsutk=b72909d000d9c7cfa8e4e0e586c49b7a&canon=https%3A%2F%2Fwww.netguru.com%2Fcodestories%2Fcertificate-pinning-in-ios&portal_id=493098&redirect_url=APefjpEtQbZTvOKc7IsZqGOsQBXfRlEXdWYR7NND6nJOyyrQf_oydfTnCnsnOzMtKb5ti3TM2zMcoex_1_jlkO8MHhOCGcq4FuuRS1tE70Z1-zCpHJ87rp60FVPZel_brAEezRuWNmvcQh6Bo4unS8D0tzdikIqNa67dv-yg11H4vR53zuNFecqYtLRXi-a-StapOTLJVNWj2HVL_AdjJ0VfdNW7Tk2-OlUV76AS7aslB_qSgEh7GnLFwvl1IvxhzSF4NDM8&__hstc=199729585.b72909d000d9c7cfa8e4e0e586c49b7a.1602436744456.1602436744456.1602436744456.1&__hss
post)



READ ALSO FROM MOBILE

Android Foldable Phones - Are You Ready?
(https://www.netguru.com/codestories/android-foldable-phones-are-you-ready)

Implement an App Store Header in React Native With Sticky Parallax Library
(https://www.netguru.com/codestories/implement-an-app-store-header-in-react-native-with-sticky-parallax-library)

On-Device Training with Core ML - Make Your Pancakes Healthy Again!
(https://www.netguru.com/codestories/on-device-training-with-core-ml-make-your-pancakes-healthy-again)

Need a successful project?

Estimate project (/estimate-project?ref=successful-project-footer-cta)

SERVICES (HTTPS://WWW.NETGURU.COM/SERVICES)

Research & Development (https://www.netguru.com/services/research-and-development)

Product Design (https://www.netguru.com/services/product-design)

UX Design (https://www.netguru.com/services/ux-design)

Branding (https://www.netguru.com/services/branding-services)

Software Development (https://www.netguru.com/services/software-development)

Web Development (https://www.netguru.com/services/web-development)

Mobile Development (https://www.netguru.com/services/mobile-development)

Digital Transformation (https://www.netguru.com/services/digital-transformation)

Machine Learning (https://www.netguru.com/services/machine-learning)

Cloud Technology (https://www.netguru.com/services/cloud-technology)

INDUSTRIES (HTTPS://WWW.NETGURU.COM/INDUSTRIES)

Fintech (https://www.netguru.com/industry/fintech)

Healthcare (https://www.netguru.com/industry/healthcare)

Retail (https://www.netguru.com/industry/retail)

Education (https://www.netguru.com/industry/education)

CUSTOMERS (HTTPS://WWW.NETGURU.COM/FEATURED-CLIENTS)

solarisBank (https://www.netguru.com/featured/solarisbank-the-first-banking-platform-for-corporations-and-startup)

Volkswagen (https://www.netguru.com/featured/volkswagen-home-first-in-the-world-concept-store-for-volkswagen)

Keller Williams (https://www.netguru.com/featured/keller-williams-digital-transformation)

Babbel (https://www.netguru.com/featured/babbel)

Damac (https://www.netguru.com/featured/damac)

IKEA (https://www.netguru.com/featured/enterprise-mobile-app-for-workspace-management)

REPORTS

Patient Management Apps (https://www.netguru.com/patient-management-apps-report)

Electric Scooter Market (https://www.netguru.com/resources/electric-scooters-report)

Banking for Families (https://www.netguru.com/state-of-banking-for-families)

Online Education (https://www.netguru.com/en/edtech-report)

Online Grocery Shopping (https://www.netguru.com/online-grocery-shopping-report)

BLOG (HTTPS://WWW.NETGURU.COM/BLOG)

Design (https://www.netguru.com/blog/topic/product-design)

Development (https://www.netguru.com/blog/topic/software-development)

Digital Transformation (https://www.netguru.com/blog/topic/digital-transformation)

Machine Learning (https://www.netguru.com/blog/topic/machine-learning)

Project Management (https://www.netguru.com/blog/topic/project-management)

Interviews (https://www.netguru.com/blog/topic/interview)

Trends (https://www.netguru.com/blog/topic/trends)

ABOUT US (HTTPS://WWW.NETGURU.COM/ABOUT-US)

Our Values (https://www.netguru.com/about-us#our-values)

How We Work (https://www.netguru.com/about-us#howWeWork)

Meet Our Team (https://www.netguru.com/about-us/team-netguru)

Press Office (https://www.netguru.com/about-us/press)

Sustainability Policies (https://www.netguru.com/sustainability)

COVID-19 Policy Statement (https://www.netguru.com/covid19-netguru-policy)

Become a Partner (https://www.netguru.com/partners/apply)

Sustainability (https://www.netguru.com/blog/topic/sustainability)

Join Our Newsletter (https://www.netguru.com/newsletter/codestories-european-tech-newsletter)

PUBLICATIONS (HTTPS://WWW.NETGURU.COM/RESOURCES)

Disruption Guide NYC 2020 (https://disruption-guide-nyc.netguru.com/)

Design Proces for Pros (https://www.netguru.com/design-process)

Project Management Guide (https://pm-guide.netguru.com/)

Scaling Fintech (https://www.netguru.com/scaling-fintech-ebook-download)

JOIN NETGURU (HTTPS://WWW.NETGURU.COM/CAREER)

Open Positions (https://www.netguru.com/career)

Careers Paths (https://www.netguru.com/career/paths)

Join as a Freelancer (https://www.netguru.com/talent/marketplace)

netguru (/)

Netguru S.A

Nowe Garbary Office Center
ul. Małe Garbary 9
61-756 Poznań, Poland

VAT-ID: PL7781454968
REGON: 300826280
KRS: 0000745671

Contact

hello@netguru.com
(mailto:hello@netguru.com)

Follow us

(https://www.behance.net/netguru)(https://dribbble.com/netguru)(https://www.facebook.com/netguru)(https://github.com/netguru)(https://pl.linkedin.com/comp

(https://clutch.co/profile/netguru)

Privacy Policy (https://www.netguru.com/tou#privacy-policy)

Terms of use (https://www.netguru.com/tou)

Sitemap (https://www.netguru.com/sitemap.xml)