Sharing Secrets between Mobile Devices

Maximilian Blochberger

blochberger@informatik.uni-hamburg.de

August 8, 2018

In this paper we show how to use the Elliptic-curve Diffie-Hellman protocol with ephemeral keys (ECDHE) in order to share a secret message between two mobile devices by using QR codes. An iOS application demonstrating this approach is presented.

1 Introduction

Sharing secrets while an attacker is eavesdropping on the process is a problem solved a while ago. Merkle [5] has proposed an idea that serves as a foundation for exchanging keys securely. Modern adoptions thereof, such as the Elliptic-curve Diffie-Hellman (ECDH), are widely used in current implementations such as the TLS protocol.

Assume that two persons want to share a secret between their mobile devices without disclosing their secret to other persons looking over their shoulders. They do not want to upload the secret to a web service, as they do not trust the service provider, and they cannot establish direct network, Bluetooth or NFC connections. The secret could then be shared by exchanging QR codes between those devices. Since QR codes are displayed on the device's screen, an observer could decode it. Password protection of the shared secret is not effective, as entering the password could also be observed. Therefore, we employ a cryptographically secure key exchange mechanism in order to protect the exchanged secret.

First, the attacker model is described. Then the process of the protected key exchange is detailed. In the end, an app is presented that demonstrates the described key exchange mechanism.

2 Attacker Model

The attacker, against whom our system is still able to protect the secret, is an outsider and has no direct access to the devices. He could be someone, who is looking at the devices in question (shoulder surfer) or he could own surveillance cameras capturing the key exchange process. He can capture and observe the screen of both devices at any given time. The attacker behaves passively and only observes the key exchange process. He is limited in his computational complexity and cannot break cryptographic systems.

3 Process

Assume that the two persons from the introduction are called Alice and Bob. The Alice wants to share a secret message m with Bob. The key exchange mechanism is basically Elliptic-curve Diffie-Hellman with ephemeral keys (ECDHE) [3, 56 pp.]. First, both of them create ephemeral key pairs, where k_A is Alice's secret and K_A Alice's public key, k_B and K_B are Bob's keys respectively. Bob first has to share his public key K_B with Alice, so that she can determine a common session secret $t = k_A K_B = k_B K_A$ from which the actual symmetric session key $k = h(t \parallel K_B \parallel K_A)$ is derived using a cryptographic hash function h. Next, she sends Bob her public key K_A as well as the encrypted message $c = E(s, m)$. Bob can now calculate the common session secret t

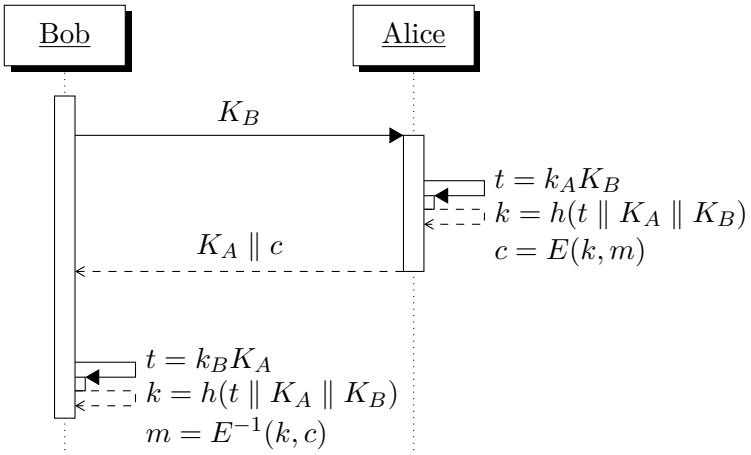

Figure 1: The process of the key exchange mechanism in detail.

and derive the symmetric key k in order to decrypt the message $m = E^{-1}(s, c)$. This requires two messages to be transmitted, which can be done be reading QR codes mutually from each others screens. The process is also depicted in figure 1.

The attacker can observe both messages but can still calculate neither t nor k since he does know neither k_A nor k_B.

4 Demonstrator

In order to demonstrate this mechanism, a demo application has been designed as depicted in figure 2. The application has a demonstration area, which allows the user to enter the message that should be shared with the other device. Obviously, the demonstration area should not be present in productive apps, as the attacker would see the shared message directly. The app works as follows:

1. Alice enters a message on her device.
2. Bob clicks *Import* on his device. A QR code containing Bob's public key K_B will be displayed there.
3. Alice clicks *Export* on her device. The camera will activate in order to scan the QR code displayed on Bob's device. The camera permission has to be granted for this. The QR code from Bob's device will automatically be detected and a QR code will be shown on Alice's device containing Alice's public key K_A and the payload c.
4. Bob clicks *Continue* on his device. The camera will active as described for Alice's device in the previous step. After the QR code from Alice's device was scanned the shared message is then displayed in the demo area on Alice's device as well.

The demo application was implemented as an open source application for iOS[1]. The Sodium crypto library[2] library is used as implementation of the key exchange mechanism, which is using X25519 [4] and Blake2b-512 [1, 2, 6] internally.

5 Limitations

This approach might be less comfortable to users than entering a password than mutually scanning screens of two devices. But even though the usability is impacted, it offers higher security with respect to the described attacker model.

Another limitation is that QR codes are limited in size. This means that the shared message m cannot be of arbitrary length.

[1] AppPETs/SecretSharing-iOS: Exchange secrets between devices using QR codes: `https://github.com/AppPETs/SecretSharing-iOS`

[2] The Sodium crypto library (libsodium): `https://libsodium.org`

Figure 2: Mockup of the user interface of the demo application.

The standard iOS SDK, which is used for scanning QR codes, does not support scanning QR codes in binary format, therefore the values of the QR codes are Base64 encoded. Due to the QR code format specification, this allows slightly more data to be packed into a single QR code.

6 Conclusion

We presented a method for exchanging secret keys of mobile devices by mutually scanning QR codes. The described approach uses state-of-the-art technology and protects against a passive attacker that could capture and observe the whole process. It works without network or Bluetooth access and can be used to quickly and securely share secrets between two devices, such as exchanging addresses without the usage of a secure messenger or encrypted mail.

Acknowledgements

This work was done in the AppPETs project[3] and supported by the BMBF.

References

[1] Jean-Philippe Aumasson et al. *BLAKE2: Simpler, Smaller, Fast as MD5*. Tech. rep. Jan. 29, 2013, pp. 1–20. URL: https://blake2.net/blake2.pdf (visited on Aug. 8, 2018).

[2] Jean-Philippe Aumasson et al. *BLAKE2X*. Tech. rep. 2016, pp. 1–4. URL: https://blake2.net/blake2x.pdf.

[3] Daniel R. L. Brown and Certicom Research. *Standards for Efficient Cryptography: SEC 1: Elliptic Curve Cryptography*. Commercial standard. Version 2.0. Standards for Efficient Cryptography Group (SECG), May 21, 2009, pp. 1–138. URL: http://www.secg.org/sec1-v2.pdf (visited on Aug. 8, 2018).

[4] Adam Langley, Mike Hamburg, and Sean Turner. "Elliptic Curves for Security." In: *RFC* 7748 (2016), pp. 1–22.

[5] Ralph C. Merkle. "Secure Communications Over Insecure Channels." In: *Commun. ACM* 21.4 (1978), pp. 294–299.

[6] Markku-Juhani O. Saarinen and Jean-Philippe Aumasson. "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)." In: *RFC* 7693 (2015), pp. 1–30.

[3] AppPETs – Datenschutzfreundliche Smartphone Anwendungen ohne Kompromisse: http://app-pets.org