Umbral: a threshold proxy re-encryption scheme

David Núñez

Abstract

This document describes the Umbral proxy re-encryption scheme used by NuCypher KMS. Umbral is a threshold proxy re-encryption scheme based on ECIES-KEM [**?**] and BBS98 [**?**].

The name "Umbral" comes from the Spanish word for "threshold".

1 Introduction

2 Preliminaries

2.1 Notation

2.2 Proxy Re-Encryption

(TODO: General description of proxy re-encryption, properties, etc)

2.3 ECIES

The Asymmetric Encryption Scheme defined in standard ANSI X9.63 [**?**], also known as Elliptic Curve Integrated Encryption Scheme (ECIES), is a hybrid encryption algorithm based on elliptic curve cryptography, symmetric encryption and message authentication codes. This algorithm is of public knowledge, and variants have been standardized also by ISO/IEC 18033-2 [**?**] and IEEE P1363A [**?**]. When producing a ciphertext with ECIES, the sender first creates an ephemeral public key and uses it for a Diffie-Hellman key agreement together with the public key of the intended recipient. The resulting shared secret is used to create the keys for the symmetric encryption and message authentication code algorithms used internally. The final ciphertext consists of the ephemeral public key, as it is necessary for decryption, and the output of the symmetric encryption and message authentication code.

A Sender can be any entity that generates data and wants to send it confidentially to a receiver, in the form of a ciphertext encrypted under the public key of the receiver.

A Receiver can be any entity that is entitled to read ciphertexts encrypted under his public key. We distinguish two types of receivers. The original receiver, which is the recipient of the data originally intended by the sender, and the delegated receiver, whom the original receiver entrusts to be able to decrypt ciphertexts initially intended to him. Therefore, there is a relation of delegation between the original receiver and the delegated one.

The Intermediary is an entity that controls the process of switching the public key of ciphertexts, from the public key of the original receiver to the public key of a delegated receiver, without being able to learn anything from the data. The Intermediary needs a key-switching key between the original and delegated receivers in order to be able to perform the key-switching process.

ECIES can be used in a variety of elliptic curves, but we will restrict the choice of elliptic curves in our cryptosystem to those that generate a group of prime order.

3 The Umbral PRE cryptosystem

In this section we present the Umbral PRE cryptosystem, defined by the following algorithms:

- $\mathsf{KeyGen}()$: Sample $x \in \mathbb{Z}_q$, compute g^x and output the keypair $(pk, sk) = (g^x, x)$.
- $\mathsf{Enc}(pk_A, M)$: On input the public key pk_A and a message $M \in \mathcal{M}$, the encryption algorithm Enc first computes $(K, epk) = \mathsf{Encapsulate}(pk_A)$, C_0 is the byte encoding of the point epk. C_1 is the ciphertext that results of applying an authenticated encryption algorithm to M with the key K. Finally, it outputs the ciphertext $C = C_0||C_1$.
- $\mathsf{Dec}(sk_A, C)$: On input the secret key sk_A and a ciphertext C, the decryption algorithm Dec first parses C as $C_0||C_1$, and decodes epk from its byte representation C_0. Next, it computes the key $K = \mathsf{Decapsulate}(sk_A, epk)$, and decrypts ciphertext C_1 using the AE decryption function to obtain message M if decryption is correct, and $\perp$ otherwise. Finally, it outputs message M (or $\perp$ if decryption was invalid).

Key Encapsulation Auxiliary Functions

- Encapsulate(pk_A): On input the public key pk_A, the encapsulation algorithm Encapsulate first generates an ephemeral keypair $(epk, esk) = (g^r, r)$, performs a Diffie-Hellman key agreement between pk_A and esk to compute a shared secret S_A, and uses this shared secret as input to a KDF to produce the encapsulation key K. Finally, it outputs (K, epk).

- Decapsulate(sk_A, epk): On input the secret key pk_A, the decapsulation algorithm Decapsulate first performs a Diffie-Hellman key agreement between epk and sk_A to compute a shared secret S_A, and uses this shared secret as input to a KDF to produce the encapsulation key K. Finally, it outputs K.

3.1 Current parameters