22:07 Timecryption, OTP with Near-polyglots

by Ange Albertini and Stefan Kölbl

Our foundation for this is the CounTeR (CTR) block cipher mode, which effectively turns a block cipher into a stream cipher. From a Nonce and a Key, it generates a keystream. The plaintext is then xored with this keystream to obtain the ciphertext. This mode acts as a one-time pad. Just an xor against a keystream, so encryption and decryption are the same operation. The cipher's decryption operation itself isn't used. If we decrypt with a different key, we end up xoring with a different keystream.

What about crafting an ambiguous ciphertext? We define this as a ciphertext that gives meaningful plaintexts for different keystreams!

To do this, recall that we can freely modify the ciphertext: the keystream is set by *(Nonce, Key)*, and plaintext and ciphertext aren't involved, which means that for a given keystream if we change ciphertext bytes, we set the plaintext bytes, as it's simply an xor against a keystream.

So, we can directly create such a ciphertext with a binary polyglot whose interpretation varies by the eky. We just independently encrypt the different ranges of the file with the different keys, then combine the two ciphertexts at the right offsets.

Making Decryption Relative to Time

But we run into a key question: how do we have an uncooperative system decrypt to two different results? We postulate that in real-world applications, specifically those having key rotation, we can do this leveraging time.

If we know the key rotation scheme used by a system, we can craft a file that, when encrypted with the current key, might be authentically decrypted later with a different key added to the key ring. (Typically, newest keys are tried first, and decrypted plaintext is returned as soon as the decryption is authenticated.) So the file will be transparently decrypted to something else, something that you decided in advance:

Timecryption combines what you want now with what you want later. You control both. When implemented against a known key rotation scheme, it's transparent and works as intended.

Near-Polyglots

Typically, each ciphertext byte belongs to one payload and one only. But if we leverage two keys from the key rotation scheme—K_1 for now and K_2 for later—we can bruteforce a nonce that will get some bytes decrypted to two different sets of values.

This means that we can make two formats that will coexist in the same file starting at offset zero, such as PDF/PE or JPG/PNG, or the same format twice, where JPG/JPG would be a near ambiguous file.

There are two ways we identify to handle these pairs of files with the same format. One way to do this is with a technique such as causing a different comment length, a bit like a hash collision for JPG/JPG. In this case, it's a file with one header and two contents. Another way is to do so for formats that work from any offset such as HTML/HTML. In this case, it's two contents coexisting in the same file.

Note that the smaller number of bytes in the overlap of the two formats, the faster the nonce bruteforce will be! The overlap only needs to be as long as is required given the specific formats. For example, *ICC* requires any parasite to start at offset 0x132, which is impractical to bruteforce. This technique can be exploited quickly with formats like JPG since it has a very small minimal offset of 4.

The Mitra repository has all the tooling for CTR, OFB and GCM modes with precomputed examples.[26]

With Authenticated Encryption

In the case of CTR encryption, it's possible to change keys because the encryption is unauthenticated, a known security risk. For this reason, the Galois/Counter Mode (GCM) was created, which is just CTR with authentication via an extra authentication data and tag. However, it's possible to forge one of the blocks such that decryption will be valid for several keys, so GCM is vulnerable too.

Secondly, more complex modes are exploitable too, such as OCB3 and GCM-SIV.[27] These cipher modes work at the block level and not at the byte level, so you need to align payloads to the block boundary. They also require more than one block to compute the authentication collision, but that's a small overhead.[28]

It's even possible to set an arbitrary content in the authentication tag!

Authenticated encryption isn't enough if the key isn't committed to the encryption. It's possible to craft ciphertexts that authentically decrypt with different keys, which is something that multiple schemes were independently found vulnerable to.[29]

Conclusion

Near-polyglots are the starting point for funky polyglot-like with cryptography, whether for AngeCryption (ECB, CBC, CFB and OFB) or Timecryption (CTR, OFB, GCM, OCB3 and GCM-SIV).

Mixing near-polyglots (CTR, OFB) and forging contents to get the same authentication tag is possible for GCM, OCB3 and GCM-SIV mode.[30]

Mitra's handling of near-polyglots makes it very easy to merge dozens of different file formats, and the *key commitments* tools forge the tags. Using these techniques and tools, exploiting authenticated collisions only requires a few command line invocations!

[26] `git clone https://github.com/corkami/mitra`

[27] `git clone https://github.com/kste/keycommitment`

[28] Note that GCM-SIV's computation cost is relative to payload size, so try it with smaller files first!

[29] `unzip pocorgtfo22.pdf project_MircoStauble.pdf %` "Actually Good Encryption? Confusing Users by Changing Nonces" by Mirco Stäuble

[30] `unzip pocorgtfo22.pdf withoutcommit.pdf`