Towards More Cryptographic Agility

Theo von Arx

March 30, 2023

Abstract

CryptPad currently uses hard coded algorithms for encrypting documents and messages. Changing such an algorithm requires the introduction of a new ciphertext type and poses a major effort on the development side.

While the currently used algorithms are still considered to be secure, the wide-spoken threat of quantum cryptography potentially endangers the security of CryptPad. Even tough there is no post-quantum public key encryption scheme standardized by NIST, it is proposed to already think ahead and plan the transition towards such a scheme. We thus show the algorithms used in CryptPad and discuss their safety against quantum computer attacks. In such a process it is key to plan it with cryptographic agility in mind. We therefore outline the path towards more easily swappable cryptographic algorithms in CryptPad.

Contents

1 Introduction

The current architecture of CryptPad uses hard-coded cryptographic primitives such as X25519-XSalsa20-Poly1305 or SHA-512 implemented in the Tweet-NaCl library for JavaScript [1], [2]. While all the used primitives are still considered to be secure and there are no known vulnerabilities, this might change in the future.

This "fear" has become more accentuated with the increasing efforts on quantum computing over the last decades. Quantum computers will be able to solve problems that are hard for today's computers. These problems include the integer factorization problem, the discrete logarithm problem, and the elliptic-curve discrete logarithm. As a consequence, the currently used primitives for asymmetric encryption and signatures would be broken and would not provide much security anymore. This danger will not only arise with the arrival of quantum computers, but already well before: companies and governments could already collect encrypted data now and decrypt it later.

Since 2016, the American National Institute of Standards and Technology (NIST) is working on standardizing Post-Quantum Cryptography. Draft standards are currently expected to be available until 2024. NIST recommends organization and companies not to wait for the standard to be done, but rather already start preparing the transition of the used primitives.

Consequently, CryptPad should already start today towards more cryptographic agility. We want to be flexible in the choice of algorithms, key sizes and other parameters. Having the possibility to more easily change the cryptographic primitives will make the transition smooth, and ensures the security of CryptPad in the long term.

2 Inventory of cryptographic primitives

Table 1 gives an overview over the used primitives. It further indicates whether the algorithms are secure against quantum computers. This is not the case for both, the asymmetric encryption and signatures algorithms in use. The reason is that the security of these is based on the elliptic discrete logarithm problem.

The most sensitive data in CryptPad is in the symmetrically encrypted login block as it contains all the keys for the user's drive, i.e., document keys, private signing and encryption keys, as well as team keys. The next-most sensitive data is then encrypted in the documents. While these are also symmetrically encrypted and thus secured in the long-term view, the problem is the key transport distribution using asymmetric encryption. A quantum attacker intercepting (or having recorded) and breaking asymmetrically encrypted messages can therefore also get access to documents.

Table 1: Overview over the used primitives in CryptPad.

Primitive	Algorithm	Usage	Quantum secure [3]
Symmetric encryption	XSalsa20-Poly1305	Login Block, Documents	Yes
Asymmetric encryption	x25519-XSalsa20-Poly1305	Messaging, symmetric key distribution	No
Asymmetric signatures	Ed25519	Proving write rights to documents and mailboxes, answering polls, access lists	No
Hashing	SHA-512	Key derivations	Yes
Key-derivation function	Scrypt	Derive login keys	Yes

Vulnerable signatures on the other hand are less severe as a quantum attacker breaking them long after it has actively modified poses only a small threat (namely to blame the author(s) on having written harmful content). However, once there exist real-world quantum computers, the signature algorithm needs to be exchanged against one that is quantum secure.

3 Criteria for updating cryptography primitives

We propose to replace the currently used cryptography primitives (i.e., XSalsa20-Poly1305, Ed25519 provided by Tweet-NaCl) when one or several of the following criteria are met:

1. There is a severe security vulnerability that cannot be fixed.
2. The NIST standardizes a post-quantum cryptography schemes, *and* this scheme has a stable and audited JavaScript implementation (or binding).
3. Cryptography operations are a restricting bottleneck for performance, *and* there are other, significantly faster libraries that provide at least the same security level.

4 Strategies

4.1 Login Block

While protecting the most sensitive data, the encryption mechanism of the login block is also comparably easier changeable than others. The only re-

quirement for cryptographic agility in the login block is that the username and password are enough to decrypt the login block.

There are two algorithms involved in the login procedure: a key derivation function (KDF) and symmetrical encryption. We therefore propose how to achieve more cryptographic agility for both of them.

Since there is no central entity knowing all the usernames and passwords, the KDF cannot be replaced from one day to another. Depending on the awareness of the users transition for all accounts will take months up to years. In case of a detected vulnerability, it is therefore key to efficiently reach all users via the internal platform, but also over email (e.g., for the emails associated to premium accounts), social media, and the blog. If a discovered vulnerability is so severe that passwords could be cracked, then users should be forced to set a new password upon login.

Change of the KDF Since the server does not have any information about the user, there is no easy way to know in advance which KDF should be used. Hence, the best way is to proceed by trial and error: check if the usage of KDF A leads to a decryptable login block. If not, proceed with KDF B, then KDF C, etc.

Unfortunately, since KDFs are required to be slow, this will take quite some time. After a successful login, the login block has to be automatically re-encrypted using the latest KDF.

Change of the symmetric encryption algorithm By putting the used algorithm into the plaintext metadata associated to a login block, we know which algorithm to use. Similar to the above, a legacy algorithm should be updated to the latest one after a successful login.

4.2 Documents

Re-encrypt under same seed One difficulty of changing the algorithms involved in documents is that we want the URL still to be valid so that users do not have to redistribute the access link. The strategy is therefore to derive new keys for a new algorithm from the same seed.

To achieve this, we specify the algorithm along with the parameters and the public verification key in the (unencrypted) metadata as well as in every sent patch. Only document owners are allowed to change the algorithm to avoid malicious user specifying broken algorithms. Also, there should be consensus on what is the best (latest) algorithm to use to avoid clients fighting about that. Currently, this can easily been done since the client code is shipped by the server and can be updated almost simultaneously.

It is up to debate whether the chanID should also be changed after algorithm change.

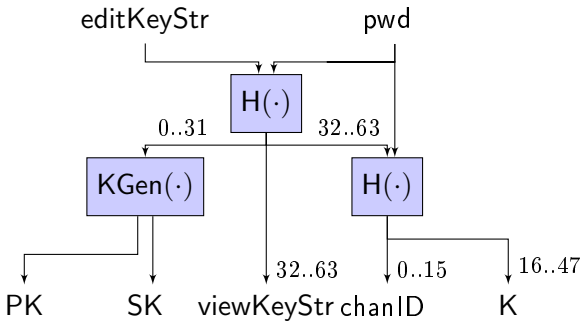

Figure 1: Key derivation for editable documents

Re-encrypt under new seed There might be the case of a severe vulnerability in SHA-512 that violates the one-way property. Attackers could consequently deduce the editKeyStr as well as the password from the viewKeyStr or even from the encrypted block if there is a second vulnerability in the encryption scheme (c.f. Fig. 1). In this case, document owner should re-encrypt the document under a new algorithm using a new, randomly generated seed. Since this involves broken links, old links should inform users that the content was changed and that they need to request the new access link.

About the seed length When planning to re-use the seed for different encryption schemes, it should be long enough to provide good entropy for future usages. While we currently use 144 Bits ($\approx$ 24 chars), 256 bits are considered to be safe beyond 2030 [4]. To keep the URLs short, we can add additional characters to the alphabet such as emojis (up to 1874 characters).

4.3 Messaging

The problem in messaging differs from the one of documents in that there is no seed to derive keys from. However, all users can pin algorithms and public encryption and verification on their profile page. A sender then automatically looks up these values and encrypts the message with the user's preferred parameters.

An unavoidable problem is that user's that did not log in for a long time cannot update their keys and algorithms. The only way to counteract leaking sensitive data over such insecure channels is to completely stop using these algorithms and thus stop messaging these users.

Another problem that already exists in the current version of CryptPad, but that is further accentuated: the profile pages are not cryptographically secured and could be spoofed.[1] A mechanism such as Signal's safety numbers can help to prevent against this.

[1] Note that this requires an active server-side attacker.

4.4 Teams

In contrast to documents, teams have a clear set of users that should be directly informed about changes of cryptographic algorithms. We can take advantage out of this and let team administrators change the encryption algorithm and distribute the new keys to all other members according to their role.

References

[1] D. J. Bernstein, B. van Gastel, W. Janssen, T. Lange, P. Schwabe, and S. Smetsers, "TweetNaCl: A Crypto Library in 100 Tweets," in *Progress in Cryptology - LATINCRYPT*, 2014.

[2] D. Chestnykhm, D. Mandiri, and AndSDev, *Tweetnacl.js*, 2016-. [Online]. Available: `https://github.com/dchest/tweetnacl-js`.

[3] T. M. Fernández-Caramès and P. Fraga-Lamas, "Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks," *IEEE Access*, 2020.

[4] D. Giry, *Cryptographic Key Length Recommendation*, 2020. [Online]. Available: `https://www.keylength.com/en/compare/`.