RÉPUBLIQUE
FRANÇAISE
Liberté
Égalité
Fraternité

ANSSI's views on crypto-agility

For developers and system architects.

1. Introduction

The past 30 years have seen some of the most used cryptographic algorithms become deprecated, and the security of systems where they are deployed become degraded. For instance, while SHA-1 certificates are uncommon today, it still took about 10 years to come to this state and it required a practical attack from researchers to motivate industrials to update their certificates. However, this is not necessarily the case for other deprecated algorithms and some are still implemented in products due to the cost of their replacement: Triple-DES is still partly in use in the banking ecosystem, and RSA-1024 was still allowed by Microsoft in late 2024.

With the upcoming threat of quantum computing, many new cryptographic algorithms have emerged, the security of which is more often than not based, for performance reasons, on variations of more conservative mathematical problems. The security impact of such variations being moderate has been a subject of studies, but remains ultimately a conjecture: vulnerabilities could potentially be uncovered in the future. Be it quantum or classical attacks, such threats make the adaptability of systems deploying such new algorithms particularly relevant, as any cryptanalysis improvements may render parameter sets or even entire cryptographic mechanisms obsolete.

The means to provide such adaptability is called crypto-agility.

Crypto-agility is the ability to extend and/or replace cryptographic components within IT-systems, without affecting their functionality, while minimising the downtime of the provided service during the update.

The term 'cryptographic component' here refers to any part of the system involved in its cryptographic functionalities. These can range from crypto libraries to hardware accelerators through secure boot or pseudo-random number generators, as well as both public and secret key material.

In practice, the introduction of new cryptographic algorithms in security products is often overlooked for various reasons, *e.g.* the need for retro-compatibility in order to ensure interoperability with cryptographic components that have not yet been upgraded. ANSSI emphasises that crypto-agility features must be considered during the benefit/risk analysis of future products or systems, as it provides a way to include these new components.

As a first remark, the ease with which crypto-agility can be incorporated heavily depends on the context, that is the system where cryptography is deployed. Namely,

the ability of the system to be updated dynamically is the cornerstone of efficient crypto-agility. For non-updatable systems with long life cycle, crypto-agility is harder to tackle, and is more limited.

Once crypto-agility options are identified, implementing them is a challenging topic and needs to be worked upon carefully. Security and complexity trade-offs are of great concern, as having many cryptographic options opens up a wider surface for attacks such as downgrade attacks, whereas switching between algorithms of various output sizes or timing performances can be a challenge to the whole IT system.

The rest of this note endeavours to detail the different levels at which crypto-agility features can be implemented as well as the challenges that each kind of component poses and, lastly, to provide some recommendations for developers to avoid security issues when implementing crypto-agility.

2. Levels of crypto-agility capabilities

As a general remark, crypto-agility requires planning in advance as in many cases, the length of the keys, ciphertexts, or signatures will grow when switching to another cryptosystem or another parameter set. Performances of the algorithms may also be impacted. All these possibilities imply designing the system with the worst case in mind, which may or may not have an impact on the performance of the system as a whole.

Parameterisation: the first level of crypto-agility consists in upgrading security parameters within a family of algorithms or even changing internal parameters within a same security level. This kind of crypto-agility is able to thwart partial security drops of algorithms, *i.e.*, whenever its bit security drops significantly, but higher levels still retain a sufficient amount of security.

Examples include but are not limited to switching from AES-128 to AES-256, from SHA-3-256 to SHA-3-512 or from ML-DSA-65 to ML-DSA-87. This also includes switching from RSA2048 to RSA4096 or from secp256r1 to brainpoolP256r1.

Algorithmic: the second level of crypto-agility is the total replacement of a cryptographic algorithm. This ensures security against the total breakdown of an algorithm or of its underlying mathematical problem. In that case, switching the cryptographic algorithm entirely is the only way to recover the security. Algorithmic crypto-agility may or may not keep compatibility with hardware accelerators for all its available algorithms.

Examples include but are not limited to switching from ML-KEM to HQC-KEM, from SHA-2 to SHA-3, from FN-DSA to ML-DSA, or from ML-DSA to SLH-DSA.

3.Implementation contexts

The capacity of a system component to modify its own cryptographic material strongly depends on its ability to be updated dynamically. As a whole, a system can be composed of multiple sub-systems, each of them offering various degrees of updatability. Hardware accelerators or crypto-processors would not be updatable while the overlying software could be. Whereas a secure boot implemented in ROM cannot be updated, the software it loads may be updated.

As a matter of fact, both of these contexts are not equally suitable for crypto-agility.

Static mechanism: These are the mechanisms which cannot be updated once embedded in a system. They can however implement multiple mechanisms from the go, and choose from them depending on a configuration file or a negotiation mechanism.

Examples include but are not limited to hardware accelerators, ICs, ROM-implemented secure boot or isolated systems in general.

Updatable mechanism: Software implemented on a connected environment is the most agile kind of system. Its cryptographic component can be updated, but crypto-agility may also be enforced similarly to static mechanisms.

Examples include but are not limited to connected software and FPGAs.

4.Recommendations

Recommendation 1. Algorithmic crypto-agility should be implemented whenever the context allows it. If not, it is recommended to use the parameter set with the highest security of a cryptographic algorithm, when such a parameter set exists. If efficiency does not allow it, only then should parameterisation crypto-agility be considered.

The "parameter set with the highest security" does not always exist for standardized algorithms: in the case of RSA, there is potentially no limit to the bitlength of the primes used, hence parameterisation crypto-agility stays meaningful in many situations. When it is the case, such as AES-128 and AES-256, safely implementing a mechanism allowing to switch from AES-128 to AES-256 may not be worth the effort compared to using AES-256 from the start.

Recommendation 2. Crypto-agility should be implemented in a consistent manner across all sub-systems of the system.

For instance, there is no point in deploying algorithmic crypto-agility on a component generating signatures if the implementation of the protocol using such signatures is unable to adapt to the new size of the signature. However, if a hardware accelerator is useless after an algorithmic crypto-agility update, then simply not using it anymore is

a way to ensure consistency across the system. In certain cases, however, a hardware accelerator may be reused across multiple algorithms.

Recommendation 3. Crypto-agility should be implemented in a way that does not break the functionality of the system.

This recommendation applies twice. If crypto-agility is implemented via updates, it applies during the issuance of such an update as well as afterwards. When switching algorithms or even simply parameter sets, new keys will be generated. In the case of short-term keys, old keys are simply discarded. In any other case, the situation is more complex. However, there exist standards describing how to handle the blend of old and new keys, such as RFC 4998 for signatures on legal documents. Afterwards, the system must be able to handle the new sizes of the cryptographic material as well as their new performances.

If crypto-agility is applied via a configuration file or a negotiation phase, the system must be able to work under any result of the negotiation phase, which means dimensioning it to the worst case scenario, in terms of cryptographic volume and timing performances.

Recommendation 4 (Updates). The integrity and authenticity protection of updates must be assured by cryptographic mechanisms in which confidence is at its highest.

Long-term security should be sought after for security of updates. Stateful or stateless hash-based signatures are the most trustful algorithms for those kinds of applications ensuring both post-quantum security and good confidence level. Signatures schemes, hybridised by combining post-quantum and classical algorithms whenever applicable, are a good alternative. Lastly, symmetric alternatives, under the form of Messages Authentication Codes to authenticate the update, can be considered with careful management of the secret keys. Update keys should not be common to multiple devices. Whatever the cryptographic means implemented, a hardware security element should be used to ensure the protection of the update key.

Recommendation 5 (Updates). Protection against replay attacks (which consist in sending an outdated update) must be implemented in the case of software updates.

Recommendations 4 and 5 are already usually considered in “traditional” software updates. One way to tackle crypto-agility via software update is to issue the crypto-agility update as a software update, and ask for a reboot.

Recommendation 6 (Configuration). if multiple algorithms coexist in a same system, the selection (or the negotiation, in the case of communication between parts of the IT system) of which exact (suite of) algorithm(s) are to be used must ensure that the process does not end up with an undesired one.

In particular compatibility with older versions of the system can cause some security issues to ensure availability by the coexistence in the same system of valid and deprecated algorithms. This security downgrade should be well restricted and deprecated cryptographic algorithms must not be allowed for new systems. Black lists of cipher suites, or at least some kind of minimal configuration (as TLS_FALLBACK_SCSV for example), should be implemented. The configuration must be protected with authentication (or at least integrity) methods and anti-replay means.

The crypto-agility provided by configuration is the only one available to static implementations, in all genericity. Namely all the algorithms that shall be used in the product must be implemented from the get-go and crypto-agility is a means to ensure the use of the desired (subset of) algorithm(s).

5.Conclusion

Non-crypto-agile systems are under threat of a major breakthrough in cryptanalysis and cannot respond effectively to the discovery of a new vulnerability in a cryptosystem it implements. Crypto-agility allows for a system to adapt itself to any threat without affecting its functionalities. Moreover, such a system may also improve its functionalities by efficiently implementing new mechanisms or state-of-the-art implementation techniques. Meanwhile crypto-agility is hard to achieve and may widen the attack surface and must be implemented carefully. A threat analysis and evaluation of the worst-case implementation must be conducted beforehand.

6.Related publications and selected bibliography

Crypto-agility is an important topic which has seen a lot of interest in the last few years. As a matter of fact, all definitions are not the same and each and any new post brings its own flavour. The definition given here is very close to the one of other European agencies such as BSI's. Some more specific points of view are available through industrial posts focusing on matters related to their own products. Lastly, some academic works give an extensive state of the art on the topic of crypto-agility.

They are talking about crypto-agility

- *Cryptographic Mechanisms: Recommendations and Key Lengths*. BSI TR-02102-1. BSI. 2024. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.html
- *Crypto-transition and Agility*. NIST. 2024. https://csrc.nist.gov/csrc/media/Presentations/2024/cryptographic-agility-and-transition-rd-and-plans/Chen-Day2-Crypto-Agility_and%2BTransition_R_and_D_Plans.pdf

- *Device security principles for manufacturers. 1. Provide updates, securely*. NCSC. https://www.ncsc.gov.uk/collection/device-security-guidance/security-principles/provide-updates-securely
- *Post-quantum cryptography: Standards and Progress*. Google. 2024. https://security.googleblog.com/2024/08/post-quantum-cryptography-standards.html
- *Crypto-agility and quantum-safe readiness*. IBM. 2024. IBM Blog Post on Crypto-agility and quantum-safe readiness (ibm.com)
- *Crypto-agility: security without compromise*. Thales. 2021. https://cpl.thalesgroup.com/sites/default/files/2021-04/cryptoagility-hse-in.pdf
- *Building Cryptographic Agility in the Financial Sector*. FS-ISAC. 2024. https://www.fsisac.com/hubfs/Knowledge/PQC/BuildingCryptographicAgilityInTheFinancialSector.pdf

Guidelines and deployment of crypto-agility

- *Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms*. RFC7696. 2015. https://www.rfc-editor.org/rfc/rfc7696
- *TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks*. RFC7507. 2015. https://www.rfc-editor.org/rfc/rfc7507
- *On the State of Crypto-Agility*. N. Alnahawi, N. Schmitt, A. Wiesmaier, A. Heinemann, T. Grasmeyer. 2023. https:/eprint.iacr.org/2023/487
- *Problems and New Approaches for Crypto-Agility in Operational Technology*. T. Frauenschläger, J. Mottok. 2024. https://eprint.iacr.org/2024/1386
- *Evidence Record Syntax* (ERS). RFC4998. 2007. https://datatracker.ietf.org/doc/html/rfc4998
- *PHOENIX: Crypto-Agile Hardware Sharing for ML-KEM and HQC*. A. Ras, A. Loiseau, M. Carmona, S. Pontié, G. Renault, B. Smith, E. Valea. 2025. http://eprint.iacr.org/2025/601