There is no Entente Cordiale in PQC yet!

Santosh Pandit

London, 21 March 2026

Important Disclaimers

All views and technical solutions presented in this paper are entirely my own and may not be necessarily shared by my current or past employers.

I have tried to explain technical issues in a simple language. Errors and omissions are therefore expected for which I assume sole responsibility.

Historical reference aims to add a touch of humour to this otherwise complex subject. If you are offended by history, just go away.

***

"Une journée sans rire est une journée perdue." – Charlie Chaplin

Contents

Executive Summary

Over the next 9 years, the Post Quantum Cryptography (PQC) transition between the UK and USA critical business counterparts will be relatively straightforward.

However, between the UK and France, the significant difference between authorities' preferences for pure and hybrid digital signatures creates risk to interoperability.

But what would happen if I were running a critical business or operation on both sides of the English channel?

For this, I have presented a temporary solution for mutual authentication; but it has its own limitations.

A modern version of Entente Cordiale is required with potentially ANSSI dropping the hard requirement for hybrid signatures. Otherwise, parties on both sides will be at risk.

Background

When I worked in the private sector, I had to learn the French language to effectively communicate with and find solutions acceptable to colleagues in multiple countries. Now I am immersed in the world of technology resilience and quantum, and once again I will try to solve a similar problem and here is its glimpse.

In two of my post-quantum laboratories (CryptoAgility.cloud and PracticallyUnhackable.com), I have already started issuing and using free digital certificates that are PQC-ready. They use NIST FIPS 204 compliant digital signatures. The technology stack works extremely well and is integrated using the latest Linux-based operating systems and OpenSSL. I have tried to see whether my current infrastructure meets regulatory requirements. Although my certificates already meet the UK and USA requirements 9 years ahead of the 2035 deadline, I find myself unable to meet the French (or German or Dutch) hard requirements on specific types of signatures. It is not my fault, actually, that the ecosystem of cryptography is not yet developed to meet regional variations. In this article, I explore the most pragmatic approach if two critical parties across the English Channel have to mutually authenticate each other to be PQC-ready.

In 1904, Britain and France signed the Entente Cordiale, a series of agreements that resolved longstanding colonial disputes and laid the foundation for diplomatic cooperation that endured through two world wars. Over a century later, these two nations find themselves needing a similar accord, this time in the domain of cryptography. The quantum computing threat to current encryption is not disputed on either side of the Channel. What is disputed is how to defend against it, and this disagreement has real consequences for businesses that operate across both jurisdictions.

What are Digital Certificates?

You and I know that when visiting any website, we check the padlock symbol in the browser, a visual representation of a certification. A digital certificate is an electronic document that proves the ownership of a public key. Think of it as a passport for a machine or a service. Just as a government issues a passport to confirm your identity, a Certificate Authority (CA) issues a digital certificate to confirm that a particular public key belongs to a particular entity, whether that entity is a web server, an email address, or an organisation.

The certificate contains the identity of the owner, the public key itself, the identity of the CA that issued it, a validity period, and a digital signature from the CA that ties all of this information together. When your browser connects to a bank over HTTPS, it receives the bank's digital certificate. Your browser then verifies the CA's signature on that certificate to confirm that the bank is who it claims to be.

Certificates form chains of trust. A root CA signs an intermediate CA's certificate, and the intermediate CA signs the end entity certificate. Your operating system and browser ship with a pre-installed set of trusted root CA certificates, forming the anchor of this chain.

The algorithms used to create these signatures and keys are the core of the problem this article addresses. Today, most certificates use RSA or Elliptic Curve Cryptography (ECC). Both are vulnerable to attack by a sufficiently powerful quantum computer running Shor's algorithm.

What are Digital Signatures?

A digital signature is the mathematical mechanism that makes certificates trustworthy[1]. It serves the same purpose as a handwritten signature on a legal document, but with far stronger guarantees: it proves who signed the data, it proves the data has not been altered since signing, and the signer cannot later deny having signed it. These three properties are called authentication, integrity, and non-repudiation.

The process works as follows. The signer takes the data to be signed and runs it through a hash function, producing a fixed size digest. The signer then encrypts this digest using their private key. The resulting value is the digital signature. Anyone with the signer's public key (published in their certificate) can decrypt the signature and compare the result to their own hash of the data. If the values match, the signature is valid.

The critical point for this article is that a digital signature is bound to a specific algorithm. A certificate signed with RSA can only be verified using RSA. A certificate signed with ML-DSA (the new post-quantum algorithm standardised as FIPS 204) can only be verified using ML-DSA. And a certificate signed with a hybrid or composite algorithm requires a verifier that understands how to process both the classical and post-quantum components together.

This last point is where the UK and France diverge.

What is Mutual Authentication?

In most everyday web browsing, authentication is one directional. When you visit a website, the server presents its certificate to prove its identity to you. Your browser verifies the certificate chain and, if satisfied, establishes an encrypted connection. At no point does the server verify your identity using a certificate.

In my view, mutual authentication will have significantly greater demand in the future for a whole range of reasons including security, sovereignty and longer certificate life.

---

[1] Note that anyone could hypothetically issue fake certificates. This is where I encourage everyone to adopt techniques including DNSSEC, DANE/TLSA and CAA and defend against MITM attacks.

Mutual authentication, also called mutual TLS or mTLS, adds the second direction. Both the client and the server present certificates to each other, and both verify the other's certificate chain. The connection is established only if both sides are satisfied.

This pattern is essential in business-to-business communications, financial services, government systems, and any scenario where both parties must cryptographically prove their identity before exchanging sensitive data. In the context of this article, imagine a UK financial institution and a French financial institution that must authenticate each other before exchanging transaction data. Both sides must present certificates. Both sides must verify the other's certificates. And both sides must do so in a manner that satisfies their respective national regulators.

This is where the trouble begins.

The Status of OpenSSL

OpenSSL is the most widely deployed cryptographic library in the world. It is the foundation of secure communications on Linux, and it underpins web servers, VPN software, email systems, and countless other applications. It is open source, battle tested over decades, and is the natural choice for any Linux based infrastructure.

As of version 3.5, OpenSSL has native support for the three NIST post-quantum standards: ML-KEM (FIPS 203) for key encapsulation, ML-DSA (FIPS 204) for digital signatures, and SLH-DSA (FIPS 205) for stateless hash-based signatures[2]. These are available in the default provider with full OID registration. My laboratory environment runs OpenSSL 3.5 on Debian Trixie, and the output of "openssl list" confirms the presence of ML-DSA-44, ML-DSA-65, ML-DSA-87, and all twelve SLH-DSA parameter sets.

This means that today, right now, I can build a complete post quantum PKI using nothing but OpenSSL on Linux. I can generate ML DSA key pairs, create certificate signing requests, issue CA certificates, sign end entity certificates, and perform TLS 1.3 handshakes authenticated with pure post quantum signatures. The technology works. It is not experimental. It is production grade cryptography in a production grade library.

What OpenSSL does not have, and has no current plan to add, is support for composite or hybrid digital signatures. There is no algorithm in OpenSSL that combines ML DSA with ECDSA into a single signature that can be placed on a certificate. You cannot issue a certificate with a composite "ML DSA 65 plus ECDSA P256" signature. The OIDs simply do not exist in the library.

The Open Quantum Safe (OQS) project previously offered an OpenSSL provider that included experimental composite signature support. However, version 0.10.0 of the OQS provider, released in mid-2025, explicitly removed all composite signature code. The project maintainers have discussed the possibility of building a standalone

[2] My other experimental laboratory (Kyber.Club) was the world’s online platform where anyone can generate FIPS 203, 204 and 205 and other key pairs completely free. Go play with it.

composite provider that delegates the actual cryptographic operations to OpenSSL's own FIPS validated implementations, but this remains a discussion on GitHub rather than shipping code.

For anyone running a Linux based infrastructure that depends on OpenSSL, this creates a hard boundary: pure PQC signatures are fully supported and work well; hybrid or composite signatures are simply not available.

The Status of Bouncy Castle

Bouncy Castle is the only widely used cryptographic library that currently implements composite signatures tracking the IETF draft specification for Composite ML DSA in X.509 certificates (draft-ietf-lamps-pq-composite-sigs). It is open source and has been maintained since its first release in May 2000. It is FIPS 140 3 validated through NIST's Cryptographic Module Validation Program. It has been used in production Java and .NET environments for over two decades.

However, Bouncy Castle is a Java and C# library. It runs on the JVM or the .NET runtime. It is not a C library. It cannot be linked against by applications that use OpenSSL. It does not integrate with the Linux cryptographic ecosystem in any native way.

For an organisation that runs a Linux-based infrastructure built on C applications and OpenSSL, adopting Bouncy Castle means introducing a Java Virtual Machine dependency, running a Java-based certificate authority such as EJBCA (Keyfactor), and maintaining two entirely separate cryptographic stacks. The certificates issued by a Bouncy Castle powered CA can contain composite signatures, but the applications that need to verify those signatures must also have access to Bouncy Castle. Your nginx web server, your OpenVPN gateway, your curl-based API client: none of these can verify a composite signature because they all depend on OpenSSL.

This is not a criticism of Bouncy Castle. It is an excellent library that serves the Java ecosystem well. The VP of Software Engineering at Bouncy Castle, David Hook, has been closely involved with the IETF composite signatures specification, and the library tracks each revision of the draft as it progresses. Proof of concept deployments using Bouncy Castle and EJBCA are already underway at several organisations.

The problem is that Bouncy Castle and OpenSSL exist in different worlds, and the European regulatory requirements for hybrid signatures currently demand a capability that only the Java world can provide. For the Linux and C ecosystem, this capability does not exist.

My Temporary Solution for Making Anglo French Business PQ Ready

Given the constraints described above, the most pragmatic approach today is the dual certificate model, supported by RFC 9763 (Related Certificates for Use in Multiple Authentications within a Protocol), published by the IETF in June 2025.

The concept is straightforward. Instead of trying to squeeze two algorithms into a single certificate (the composite approach), each entity holds two separate certificates: one classical (for example, ECDSA P256) and one post quantum (for example, ML DSA 65). RFC 9763 defines a new X.509 extension called RelatedCertificate and a corresponding CSR attribute called relatedCertRequest. These provide cryptographic binding between the two certificates, proving that the same entity controls both private keys. This binding is achieved by signing a proof of possession of the first certificate's private key during the CSR process for the second certificate.

For the mutual authentication scenario between a UK party and a French party, the protocol works as follows.

Both parties hold two certificates each, one classical and one PQC, linked via the RFC 9763 RelatedCertificate extension. The four certificates in total can all be issued using OpenSSL on Linux, since OpenSSL fully supports both ECDSA and ML DSA as standalone algorithms.

During mutual authentication, both certificates are presented and verified. The French party's regulator (ANSSI) requires assurance from both a classical and a post quantum signature. By verifying the classical certificate and the PQC certificate independently, and confirming their binding through the RelatedCertificate extension, the French party achieves the "both algorithms must hold" property that ANSSI mandates for hybrid protection.

The UK party's regulator (NCSC) prefers pure PQC. The UK party can verify just the PQC certificate if it chooses, or both if it wishes. Either way, the NCSC requirement is satisfied because a PQC signature has been verified.

The principal limitation of this approach is at the TLS protocol layer. Current TLS 1.3 (RFC 8446) supports only a single certificate chain per authentication. There is an IETF draft (draft-yusef-tls-pqt-dual-certs) that extends TLS 1.3 to negotiate and present two certificate chains in a single handshake, but it is not yet adopted by the TLS working group and is not implemented in any shipping TLS library.

Until the dual certificate TLS extension is standardised and implemented in OpenSSL, the workaround is to handle the dual verification at the application layer. The TLS connection is established using one certificate (either classical or PQC, depending on the negotiated preference), and the second certificate's signature verification is performed within the application protocol. This is not elegant, but it works, and it works entirely within the OpenSSL and Linux ecosystem.

I consider this a temporary solution because it requires application-level awareness of the dual certificate requirement. It places a burden on application developers rather than being handled transparently at the transport layer. It will become unnecessary once the TLS protocol catches up with the PKI layer.

A Modern PQC Entente Cordiale is Still Needed

The technical workaround described above is just that: a workaround. It does not resolve the fundamental regulatory divergence that creates the problem in the first place.

The UK's NCSC and the USA's NSA (through CNSA 2.0) both prefer pure PQC signatures. Their reasoning is sound. Digital signatures do not face the "harvest now, decrypt later" threat that makes hybrid key exchange a reasonable precaution for encryption. A signature is verified at the time it is used. If the algorithm was secure at that moment, the authentication was valid. There is no stored signature to be forged retroactively.

France's ANSSI, Germany's BSI, and the Netherlands' NLNCSA take the opposite position. They require hybrid signatures (classical plus PQC combined) for all public key operations, with an exception only for hash-based signature schemes like SLH DSA and LMS. Their reasoning is also defensible: ML DSA is built on structured lattice mathematics that has been studied for far less time than RSA or ECC, and the side channel attack surface of implementations is not yet fully characterised. Hybrid signatures provide a safety net if the new algorithm proves flawed.

Both positions have merit. But together, they create a situation where no single certificate can satisfy regulators on both sides of the Channel. The composite signature approach that would satisfy ANSSI is not supported by the most widely deployed cryptographic library in the world. The pure PQC approach that satisfies NCSC and NSA is dismissed by BSI and ANSSI as insufficient.

This is not a technical problem waiting for engineers like me to solve. The technology for pure PQC certificates exists today and works. The technology for composite certificates exists in a different ecosystem (Bouncy Castle) and is approaching standardisation at the IETF. The problem is that regulatory authorities have not coordinated their requirements in a way that allows a single, practical solution to satisfy all of them simultaneously.

What is needed is a diplomatic and regulatory alignment, a modern PQC Entente Cordiale, where the UK, France, Germany, the Netherlands, and the USA agree on a transition approach that industry can actually implement with the tools that exist. The most natural resolution would be for the European authorities to accept pure PQC signatures as sufficient for mutual authentication, at least as an interim measure, while the composite signature ecosystem matures. ANSSI already exempts hash-based signatures from the hybrid requirement, acknowledging that well understood mathematical foundations reduce the need for a classical safety net.

The structured lattice problems underlying ML-DSA have been studied for over two decades and survived three rounds of NIST evaluation involving the global cryptographic community. Extending a similar level of trust to ML-DSA, even provisionally, would unblock cross-border PQC deployment today rather than in several years' time.

Without such alignment, businesses operating across the Channel will be forced to maintain parallel cryptographic stacks, implement application layer workarounds, and accept that the PQC transition, already a complex multi-year migration, is made significantly harder by regulatory fragmentation than by any technical limitation.

The original Entente Cordiale of 1904 succeeded because both nations recognised that their disputes, while real, were less important than the benefits of cooperation. The same calculus applies here. The quantum threat is coming for everyone. A coordinated response will serve both nations far better than divergent requirements that slow down the very migration they intend to accelerate.

Santosh Pandit

London, 21 March 2026

References

Note: All links were working when I checked on 21 March 2026. If they stop working, please use the Internet Archive or 404wayback or contact the respective authors.

1. NIST FIPS 203, Module Lattice Based Key Encapsulation Mechanism Standard (ML KEM), August 2024.
2. NIST FIPS 204, Module Lattice Based Digital Signature Algorithm (ML DSA), August 2024.
3. NIST FIPS 205, Stateless Hash Based Digital Signature Algorithm (SLH DSA), August 2024.
4. NIST IR 8547 (Initial Public Draft), Transition to Post Quantum Cryptography Standards, November 2024. (Note: As the public comment period has closed, this same page should have a link to the final version in the future)
5. NIST SP 800 227, Recommendations for Key Encapsulation Mechanisms, September 2025.
6. NSA, The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ, Version 2.1, December 2024. (Note: Press Ctrl and click link as this is a pdf file. If you want the latest from the NSA, visit this NIST page.)
7. NCSC UK, Next Steps in Preparing for Post Quantum Cryptography, Version 2.0. (Note: The bottom of this page has links to other useful NCSC UK articles).
8. ANSSI, ANSSI Views on the Post Quantum Cryptography Transition, 2023 (updated position). (Note: Press Ctrl and click link as this is a pdf file. Another useful ANSSI document in English on cryptoagility can be found here.)
9. BSI, Technical Guideline TR 02102 1, January 2026. (Note: Please use the link on this page to access the pdf file in English. Also, a side note in case you actually read the documents. The BSI requirements are somewhat different to that in the UK and USA e.g. the use of Brainpool Curves. I have removed those algorithms from my infrastructure as the BSI supports other curves too).
10. Joint Statement: Securing Tomorrow, Today. (Note: I could be wrong, but I think this was initiated by Germany, France, and the Netherlands. The webpage says it is signed by government authorities from 18 EU member states. Use the link on the page for pdf in English)

(Special note about the RFC / IETF citations below: I have included links to the IETF tracker pages. Each document will have the link to its final version in due course, subject to standard IETF governance.)

11. RFC 9763, Related Certificates for Use in Multiple Authentications within a Protocol, A. Becker, R. Guthrie, M. Jenkins, June 2025.

12. RFC 9794, Terminology for Post Quantum Traditional Hybrid Schemes, IETF, 2025.
13. draft-ietf-lamps-pq-composite-sigs-15, Composite ML DSA for use in X.509 Public Key Infrastructure, M. Ounsworth, J. Gray, M. Pala, J. Klaussner, S. Fluhrer, February 2026.
14. draft-yusef-tls-pqt-dual-certs-01, Post Quantum Traditional Hybrid Authentication with Dual Certificates in TLS 1.3, December 2025. (Note: I would personally like to see this paper getting traction).
15. draft-reddy-pquip-pqc-signature-migration-01, Guidance for Migration to Composite, Dual, or PQC Authentication, October 2025.
16. International PQC Requirements, Post Quantum Cryptography Coalition (PQCC), May 2025. (Note: I suggest you bookmark their main page, which has excellent information)
17. OpenSSL 3.5, https://www.openssl.org. (Note: If you want to verify the latest versions available use the downloads page instead.)
18. Bouncy Castle Cryptographic APIs, bouncycastle.org website.
19. Open Quantum Safe Project, website is here and GitHub is here.
20. Keyfactor EJBCA, website is here.