Why One Time License Signing Matters for Remote PKI Workflows

When I first started working with PKI deployments, the word "license" meant a static sticker on a server rack. You bought the hardware, you installed the software, and you forgot about it until renewal season. But over the past few years, that model has cracked open. Teams now sign documents and code from anywhere, often using the same USB tokens they already own, and the licensing model around that signing has shifted in ways that are easy to miss.

Let me walk through what I have seen in the field, specifically around one time license signing, how it works for remote signing, and why it matters for anyone managing digital certificates, HSMs, or cloud PKI today.

The Old Way: Per-Seat and Per-Token Licensing

For a long time, a PKI deployment meant buying a license for every token. If your team had twenty YubiKeys or SafeNet eTokens, you paid for twenty licenses. If you wanted to enable PDF signing on a laptop, you installed a local client and tied the license to that machine. The model worked, but only if your team never moved, never shared tokens, and never needed to sign from a hotel room or a client site.

That is where the friction started. A developer on my team once needed to sign a code update with a Code Signing Certificate stored on a USB Cryptographic Token. She was working remotely, and the local license on her laptop had expired. We had to courier a new license file to her, which took two days. The signing itself took three seconds. That imbalance made no sense.

What One Time License Signing Actually Means

One time license signing is not about buying a perpetual license. It is about a licensing event that happens once per token or once per signing session, rather than per seat or per year. In practice, it means your team can plug a USB token into any machine that has the right signing software, authorise the license once, and then sign documents or code without worrying about license servers, activation keys, or expiry dates tied to a specific device.

I have seen this model used most effectively with remote signing setups. A team stores private keys on an HSM or a hardware security module in a data centre, but the signing request comes from a laptop in a co-working space. The license check happens one time, the token authenticates, and the signature is applied. After that, the license does not need to be on that laptop again unless the token is removed and reinserted.

How It Fits Into Remote Signing

Remote signing itself is not new, but the licensing around it has been a bottleneck. If your organisation uses Cloud PKI, the signing server might be in one country and the signer in another. Traditional licensing models assume the signer and the license are on the same network. That assumption breaks down fast.

With one time license signing, the license is attached to the cryptographic token, not to the network. You authorise the token once, and from that moment the token can sign from any location, as long as the remote signing service can reach it. This is especially useful for teams using YubiKeys or SafeNet eTokens for PDF Signing, or for developers who need to sign code with Microsoft Authenticode.

Practical Example: Code Signing Across Time Zones

I worked with a small software firm that had three developers spread across Singapore, Berlin, and San Francisco. They all needed access to the same Code Signing Certificate stored on an HSM in a Singapore data centre. The traditional licensing model would have required three separate licenses, one per developer, with a renewal every year. Instead, they used a remote signing platform that supported one time license signing. Each developer plugged in their own USB Cryptographic Token, authenticated once, and the license was tied to that token. No renewal, no per-seat cost, no license server to maintain.

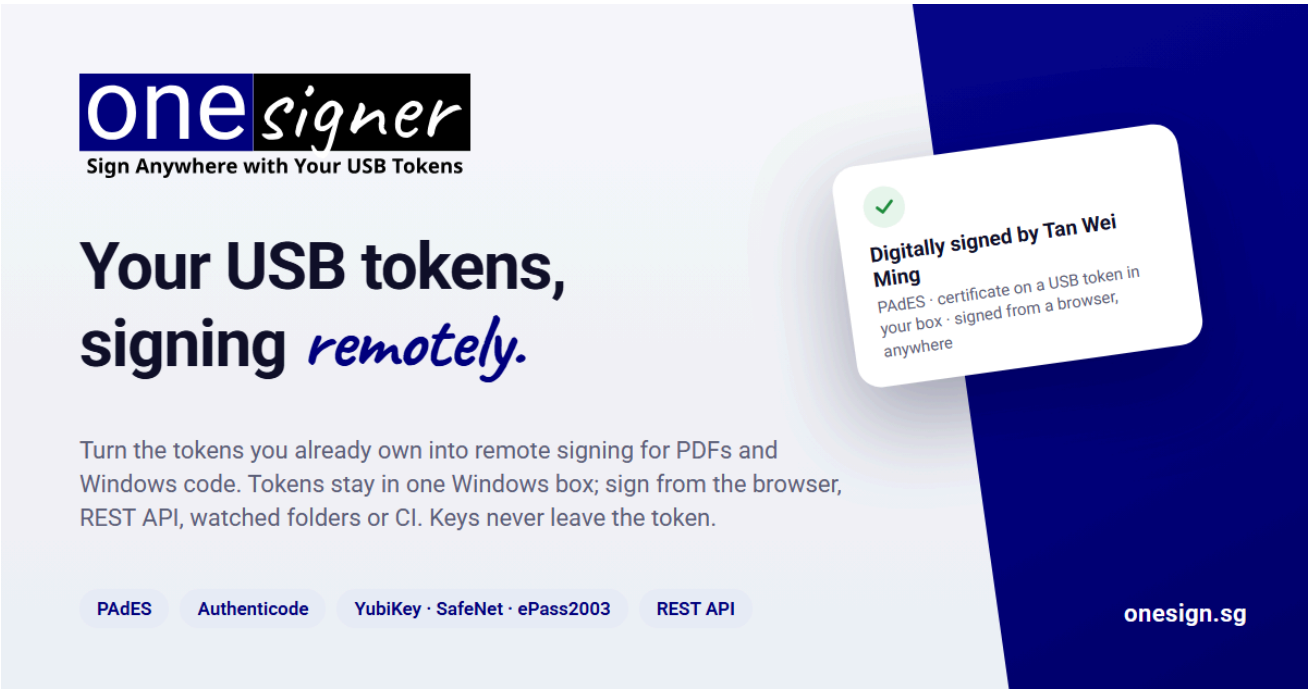
That approach saved them roughly sixty percent on licensing over two years, but more importantly, it removed the friction of license management from their release pipeline. When a critical patch needed signing on a Sunday afternoon, nobody had to check whether the license was still valid on that particular laptop.

Regulatory Considerations: eIDAS and the Adobe Approved Trust List

When you talk about digital signatures in Europe, eIDAS compliance is unavoidable. The regulation requires that Qualified Certificates be stored on secure devices, typically USB tokens or HSMs, and that the signing process maintain a clear audit trail. One time license signing does not conflict with eIDAS, as long as the license itself does not bypass the security of the token. The license authorises the software to use the token, not the token itself. The private key stays on the hardware, which is exactly what eIDAS requires.

Similarly, if you are signing PDFs that need to be trusted by Adobe readers, the certificate must appear on the Adobe Approved Trust List. The licensing model around that certificate has no effect on trust list membership. What matters is that the signing software is properly licensed and that the audit trail captures the signing event. One time license signing makes it easier to

maintain that audit trail because the license event is recorded once and does not clutter logs with renewal or reauthorisation entries.



Hardware Compatibility: YubiKey, SafeNet eToken, RSA SecurID

Not all USB tokens behave the same way under a one time license model. I have tested this with several form factors. YubiKeys work well because they support a standard PKCS#11 interface that most signing software can use without custom drivers. SafeNet eTokens are similar, though some older models require a middleware layer that can complicate the license handshake. RSA SecurID tokens, which are primarily used for authentication rather than signing, can be used in some configurations, but they are not ideal for long-term certificate storage.

The key point is that one time license signing works best when the token exposes a standard cryptographic interface. If your team is already using USB Cryptographic Tokens for PKI, you likely do not need to buy new hardware. You just need signing software that supports a one-time license model.

Trade-Offs and When This Model Does Not Fit

I would be misleading if I said one time license signing is always the right answer. It has trade-offs. First, if you lose the token, you lose the license. Unlike a subscription model where you can reissue a license from a server, the one-time model ties the license to physical hardware. That means you need a backup plan for lost or damaged tokens.

Second, some audit requirements expect a license to be tied to a specific user or machine, not to a token. In highly regulated environments like finance or healthcare, the license itself may need to be part of the audit trail. A token-based license can still satisfy that, but you need to log which token was used and who used it, not just that a license was present.

Third, if your team uses an HSM that is shared across dozens of users, a per-token license might not scale well. In that case, a server-based license for the HSM itself, combined with user authentication, often makes more sense. One time license signing is best for teams where each user has their own token and needs to sign from multiple locations.

Practical Steps for Implementing One Time License Signing

If you are considering this model, here is what I have found works in practice:

- Choose signing software that explicitly supports token-based licensing. Not all remote signing platforms do. Look for one that ties the license to the token serial number or to the certificate on the token.
- Test with your actual hardware. A YubiKey may work differently than a SafeNet eToken, even if both are USB tokens. Run a full signing workflow, including license authorisation, before rolling out to the team.
- Document the license activation process. Since the license is tied to the token, a new team member needs to know how to authorise their token. A simple internal guide saves time later.
- Set a reminder to revoke licenses when a token is decommissioned. The one-time model does not automatically expire, so you need a process to remove old tokens from your license tracking system.

The Bigger Picture: Digital Workspace and Cloud PKI

Looking ahead, the trend is toward digital workspaces where signing can happen from any device, not just a company-issued laptop. Cloud PKI is part of that shift, but licensing has lagged behind. One time license signing bridges the gap between the flexibility of remote signing and the control of hardware-based security.

I have seen teams adopt this model and then stop thinking about license management entirely. That is the real win. When the license fades into the background, the signing workflow becomes as simple as plugging in a token and clicking sign. No license server, no activation window, no renewal reminder. Just one time license signing, and done.

For teams looking at remote signing solutions like OnesSign, the licensing model is worth examining before you commit. A model that ties the license to the token rather than the seat

gives you more flexibility, especially if your team works across locations and time zones. It also aligns well with the way modern PKI works: the security is in the hardware, and the license should support that, not fight it.