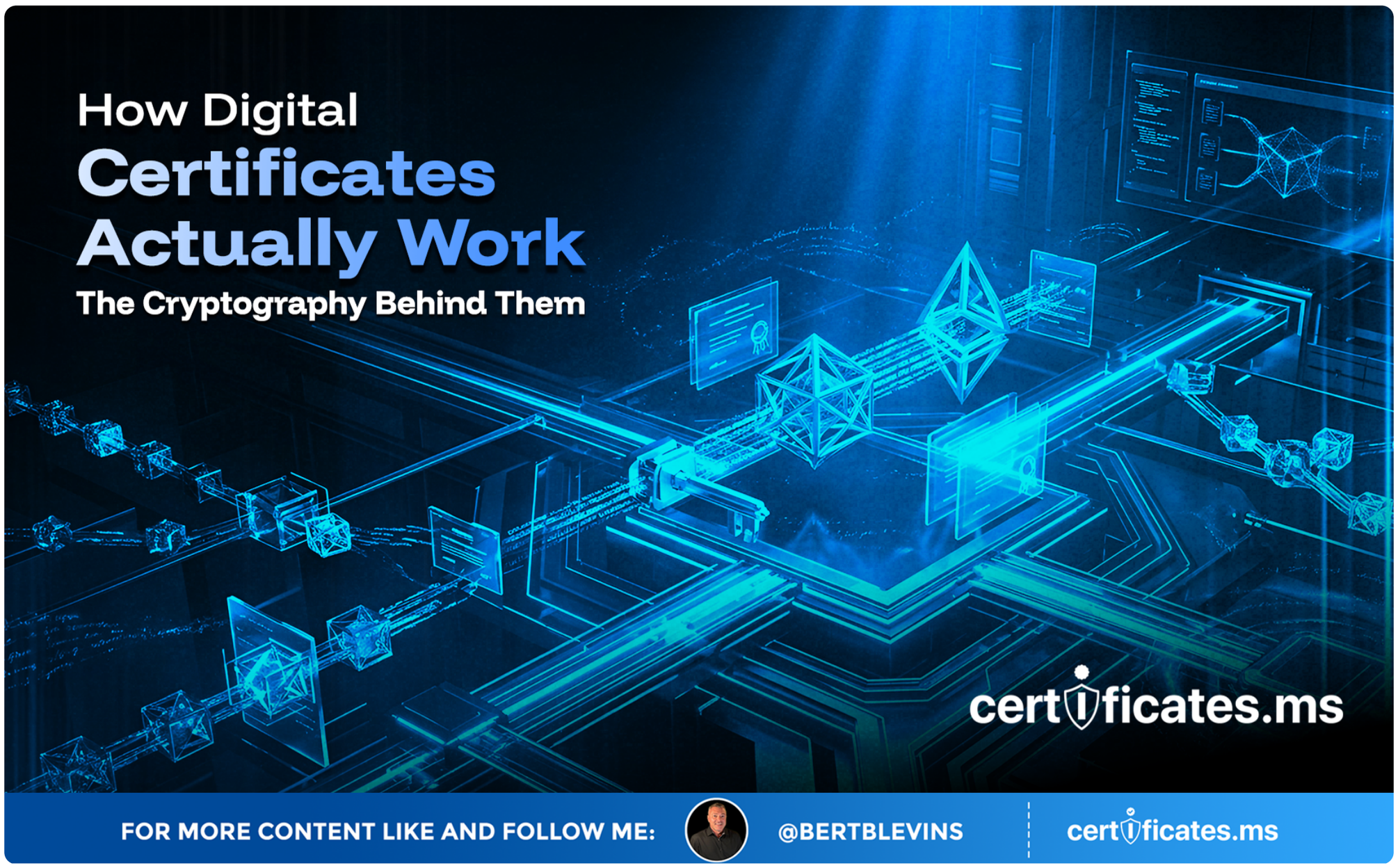
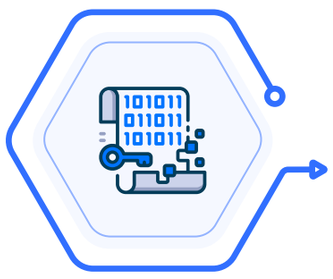
How Digital Certificates Actually Work:
The Cryptography Behind Them



Certificates get discussed constantly in terms of what they do: encrypt traffic, prove identity, unlock the padlock icon. Far less often does anyone explain the actual mathematics making any of that possible. This article opens the hood on the cryptography behind digital certificates, translated into terms that do not require a mathematics degree to follow.

Two Keys, One Relationship

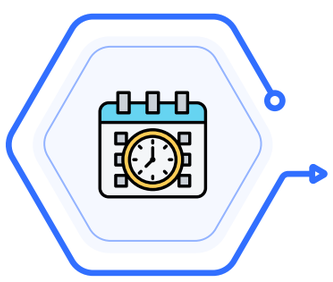
Certificates rely on asymmetric cryptography, sometimes called public key cryptography. Unlike a traditional password or a shared secret, asymmetric cryptography uses two mathematically linked keys: a public key that can be shared with anyone, and a private key that must never be shared with anyone. The relationship between the two is what makes the whole system work. Data encrypted with the public key can only be decrypted with the matching private key. A signature created with the private key can be verified by anyone holding the public key, without that verifier ever needing access to the private key itself.



This asymmetry solves a problem symmetric encryption cannot: how do two parties who have never met establish a secure channel without first exchanging a secret over an insecure medium? With asymmetric cryptography, the public key can be shouted from the rooftops. Only the corresponding private key can do anything meaningful with data protected by it.

The Math Underneath, in Plain Terms

Most certificates today rely on one of two mathematical foundations. RSA relies on the practical difficulty of factoring the product of two very large prime numbers; multiplying two large primes together is fast, but working backward from the product to find the original primes is, with current computing power, prohibitively slow. Elliptic Curve Cryptography, increasingly the preferred choice for new deployments, relies on a different hard problem involving points on a specially defined curve, and achieves equivalent security to RSA with dramatically smaller key sizes, which translates into faster handshakes and less bandwidth overhead.



Neither system is unbreakable in an absolute sense. Both rely on problems that are computationally infeasible to solve with classical computers in any practical timeframe, given sufficiently large key sizes. That distinction, computationally infeasible rather than mathematically impossible, matters, and it is exactly why key sizes and algorithm choices get revisited periodically as computing power advances.

How Signing Actually Proves Something

When a Certificate Authority signs a certificate, it is not simply attaching a stamp of approval. It runs the certificate's contents through a cryptographic hash function, producing a fixed-length digest unique to that exact content, then encrypts that digest with its own private key. Anyone holding the CA's public key can reverse that step, decrypt the digest, independently hash the certificate's contents themselves, and compare the two. If they match, two things are proven simultaneously: the certificate has not been altered since signing, and it was genuinely signed by whoever holds that CA's private key. Change a single character in the certificate and the hash changes completely, making tampering immediately detectable.

Symmetric Encryption Does the Heavy Lifting

Asymmetric cryptography is computationally expensive relative to symmetric encryption, so it is used sparingly. During a TLS handshake, asymmetric cryptography's job is limited to authentication and to securely negotiating a shared symmetric key. Once that shared key is established, the actual bulk data, the webpage content, the API payload, the file transfer, is encrypted using fast symmetric algorithms like AES. This hybrid approach captures the security benefits of asymmetric cryptography's identity verification while keeping performance practical for high-volume traffic.

Why Key Length and Algorithm Choice Matter

A certificate's security is only as strong as the weakest link in this cryptographic chain: the algorithm, the key length, and the hash function all matter. Older hash functions like MD5 and SHA-1 have known weaknesses and have been phased out of certificate issuance industry-wide. Key lengths that were considered secure a decade ago are now considered inadequate as computing power has grown. This is why certificate standards evolve, and why an organization running certificates with outdated cryptographic parameters is carrying real, measurable risk even if nothing has visibly gone wrong yet.

The Cryptography Behind AI-to-AI and AI-to-Service Traffic

Every principle covered above applies identically whether the connection being secured is a human loading a webpage or an AI agent calling an internal API, retrieving a document, or coordinating with another agent. The cryptography does not know or care what kind of entity is on either end of the handshake; it only cares whether the keys and signatures check out. What has changed is volume and velocity. AI-driven systems generate and consume far more connections per second than human-driven traffic ever did, which means the underlying cryptographic operations, key generation, signing, and verification, now run at a scale that makes efficient algorithms like Elliptic Curve Cryptography increasingly attractive over the more computationally heavy RSA for new AI infrastructure deployments.

The Countdown Is Already Running: 200 Days, 100 Days, 47 Days

Every certificate conversation in 2026 eventually arrives at the same clock, and it is worth closing on it here. The CA/Browser Forum's Ballot SC-081v3 is not a proposal under discussion; it is an approved, already-in-motion schedule. Maximum public TLS certificate lifetimes fall from 398 days to 200 days on March 15, 2026. They fall again to 100 days on March 15, 2027. By March 15, 2029, they drop to just 47 days, with domain validation itself needing to be re-proven roughly every 10 days.

01

Translate that into operational terms and the picture gets stark quickly. An organization currently renewing certificates a few times a year will be handling renewal events on the order of every couple of weeks by the end of this countdown, across every endpoint it operates. Manual tracking, calendar reminders, and a spreadsheet somebody checks once a month will not survive contact with that cadence. What has always been an occasional chore is becoming a continuous, automated operation, whether an organization plans for it or not.

02

Understanding the cryptography above matters more, not less, as certificates get reissued every 200, then 100, then eventually 47 days, since each reissuance means the keys, signatures, and algorithms described here are being generated and verified all over again, at a pace only automation can sustain.

03

The 200-day, 100-day, and 47-day milestones are not distant hypotheticals; the first has already arrived. Organizations that build the automation loop now, generating keys, vaulting them securely, brokering issuance across Certificate Authorities through APIs, and rebinding certificates to live endpoints without manual intervention, will meet each deadline without disruption. Organizations that wait will be rebuilding their certificate operations under deadline pressure, with far less room for error and far less time to get it right. The countdown is the call to action. The only real decision left is whether to automate on your own schedule, or on the CA/Browser Forum's.