Darwinium

SOLUTION BRIEF

Data Privacy & Sovereignty

A fraud-prevention platform engineered so that **it cannot read your customers' data**, and neither can anyone who breaches it. PII is encrypted inside your own perimeter, stored in your own cloud, and decrypted only inside an authorized analyst's browser. It never exists in the clear on our systems, even for a moment.

FOR SECURITY & RISK LEADERS

THE INEVITABLE BREACH

You can't leak what you can't read

Every vendor you onboard becomes part of your attack surface. And the data now says that surface is where breaches happen:

- **The average data breach costs $4.44M globally, a record $10.22M in the US.**[1]
- **The average breach goes undetected for 241 days**, roughly eight months of silent exposure.[1]
- **Nearly half of all breaches now originate with a third party**, up from ~30% to ~48% in a single year, now the fastest-growing breach vector.[2]
- **"Assume breach" is now formal guidance**: NIST SP 800-207[3] tells enterprises to assume an attacker is already inside.

So the honest question for any security leader evaluating a data-handling vendor is no longer *"will they get breached?"* It's *"when they are, is my customers' PII in the blast radius?"* For almost every SaaS vendor, the answer is yes: your customers' names, emails, phone numbers and card data sit in their database, decryptable by their own systems, one stolen credential away from the front page.

> THE PATTERN EVERY SECURITY LEADER NOW RECOGNIZES
>
> A single **file-transfer** vulnerability cascaded to **2,700+ organizations and ~93 million people.**
>
> At a major **cloud data platform**, **~165 customer environments** were breached via stolen credentials, *while the platform itself was never compromised.* Where the data lived, and who could decrypt it, mattered more than the vendor's perimeter.
>
> A leading **identity provider's** support-system compromise reached **134 of its customers** through stolen session tokens, hitting even security-elite firms.

The takeaway isn't "pick more careful vendors." It's that **data minimization is the only security control that fails safe.** If a vendor never holds decryptable PII, breaching that vendor yields ciphertext, not identities, and the blast radius collapses to encrypted blobs sitting on infrastructure *you* own.

Who Darwinium is

Darwinium is a **continuous security and fraud-prevention platform** that runs at the edge of your own infrastructure, inside the CDN you already operate (Cloudflare, AWS CloudFront or Akamai). It protects your customer journeys from fraud, scams, bots and account abuse, in real time, across web, mobile and API. To do that job well, Darwinium must reason over some of the most sensitive data you hold. **So we built the platform so that we never see any of it in the clear.**

~48%

of data breaches in 2026 involved a **third party**, nearly double the year before, and now the fastest-growing way organizations get breached.

VERIZON DBIR · 2026[2]

The control question has shifted.

THE OLD QUESTION

"How do we keep attackers out of our vendors' systems?"

THE QUESTION THAT MATTERS NOW

"When a vendor is breached, and one will be, can anyone actually read our customers' data?"

Most vendors answer the first question with promises. Darwinium answers the second by construction. We deliberately **over-engineered privacy** so that the platform protecting your customers can never become a liability holding their data:

- **We encrypt PII inside your perimeter**, before it ever reaches us.
- **We store it in your own cloud**, your bucket, your region, your control.
- **We decrypt it only in the browser**, never on a Darwinium server.
- **We keep the keys away from our own machines**; decryption requires a human with permission.

The pages that follow walk through each layer, and what an attacker gets if they breach us anyway. *(Spoiler: nothing they can read.)*

01 THE CAPABILITY

What ”over-engineered privacy” actually means

Most vendors describe privacy as "encrypted at rest and in transit." That's table stakes, and it's *vendor-decryptable* encryption: the keys sit next to the data, so a deep enough compromise reads both. Darwinium's model is different in kind. It rests on four design decisions, each shrinking what an attacker, or a rogue insider, or a legal demand served on *us*, can obtain.

1 Encrypt inside your perimeter, before we receive it

Darwinium runs as lightweight code inside your own CDN edge. The moment sensitive data is captured, it's encrypted **on your edge, inside your perimeter, before anything is sent to Darwinium.** The scheme is **Hybrid Public Key Encryption (HPKE), RFC 9180**[4], the modern standard that also underpins TLS Encrypted Client Hello. It encrypts to a **public** key at the edge, while the matching **private** key stays locked away.

```
KEM   :  DHKEM(P-256, HKDF-SHA256)     // NIST P-256 key encapsulation
KDF   :  HKDF-SHA256
AEAD  :  AES-256-GCM                    // authenticated payload encryption
```

WHY THIS MATTERS

Because encryption uses a **public** key, the edge protects data with no ability to *decrypt* it, and every record gets a fresh, single-use key. A compromised edge cannot read past traffic: the split-role guarantee of public-key cryptography, applied at the moment of capture.

2 Make it searchable without making it readable

Fraud investigation needs search. Storing the raw email would defeat the purpose, so each identifier is transformed by a **one-way elliptic-curve process** (SHA-256 hashed, mapped onto a Curve25519 point, *ristretto*-encoded). It supports exact-match search but is **irreversible and rainbow-table resistant.** Only this anonymized token reaches Darwinium, never the raw value.

3 Store it in your cloud; you own the data

Encrypted PII is written **directly from your edge into a storage bucket you own and control** ("bring your own bucket"). The cleartext never touches Darwinium's SaaS. Access splits into a **write-only** role (the edge) and a **read-only** role (the Portal), unlockable *only by a human.* The bucket is **S3-compatible** (AWS S3, Azure Blob or Cloudflare R2), in **any region you choose**, and you can read your own data **without calling Darwinium at all.**

4 Decrypt only in the browser, a zero-knowledge read path

When an authorized analyst views PII, decryption happens **inside their browser**, never on a Darwinium server. This is **In-Browser Decryption**; the next two pages are the deep dive.

FIGURE · THE WRITE PATH

Encryption, inside your perimeter

Nothing sensitive leaves your environment in readable form. On your own CDN edge, each PII value is encrypted to a public key, and each identifier is separately transformed into an irreversible, searchable token. Two outputs diverge from there: **encrypted PII goes straight to your bucket**; **only an anonymized index goes to Darwinium.**

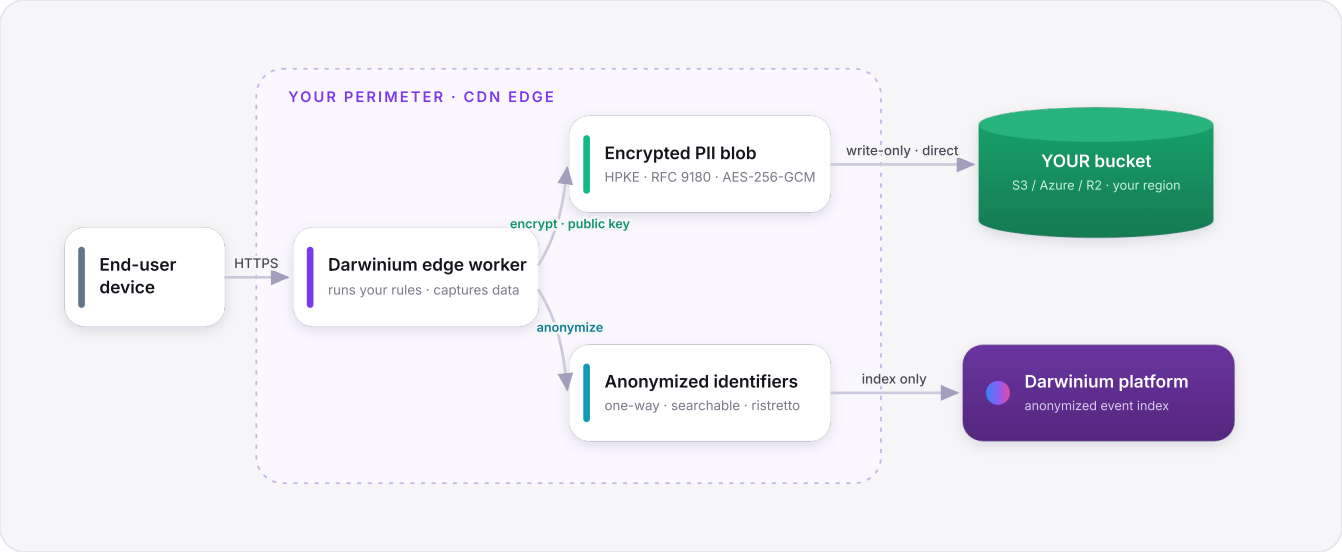


FIG 1 Sensitive data is encrypted on your own CDN edge, inside your perimeter, before anything reaches Darwinium. Encrypted PII is written directly to your bucket with a write-only role; only a one-way anonymized index goes to Darwinium.

GOES TO YOUR CLOUD

Encrypted PII blob, HPKE-sealed to your public key, a fresh single-use key per record, written directly to a bucket you own in a region you choose.

GOES TO DARWINIUM

Anonymized index only, one-way hashed identifiers that make events searchable by exact value while remaining irreversible. No raw PII, ever.

The vendor that can't read your data

Conventional "secure" tooling decrypts on the server and sends plaintext to the screen, so the vendor's servers hold your customers' PII in memory on every single investigation. Darwinium removes that step. When an authorized analyst opens a record:

① The browser proves the analyst holds a **signed "Read PII" permission** and requests the record.

②–④ The encrypted record is fetched **directly from your bucket, in its own region**, never routed through Darwinium's core. Each record is two parts: a tiny *encapsulated key* and the *encrypted body.*

⑤ The browser sends **only the encapsulated key** (a few bytes of key material, *not* the data) to a key service.

⑥ The service returns a **single-use key for that one record**, and nothing else.

⑦ The browser **decrypts the body locally** and displays it.

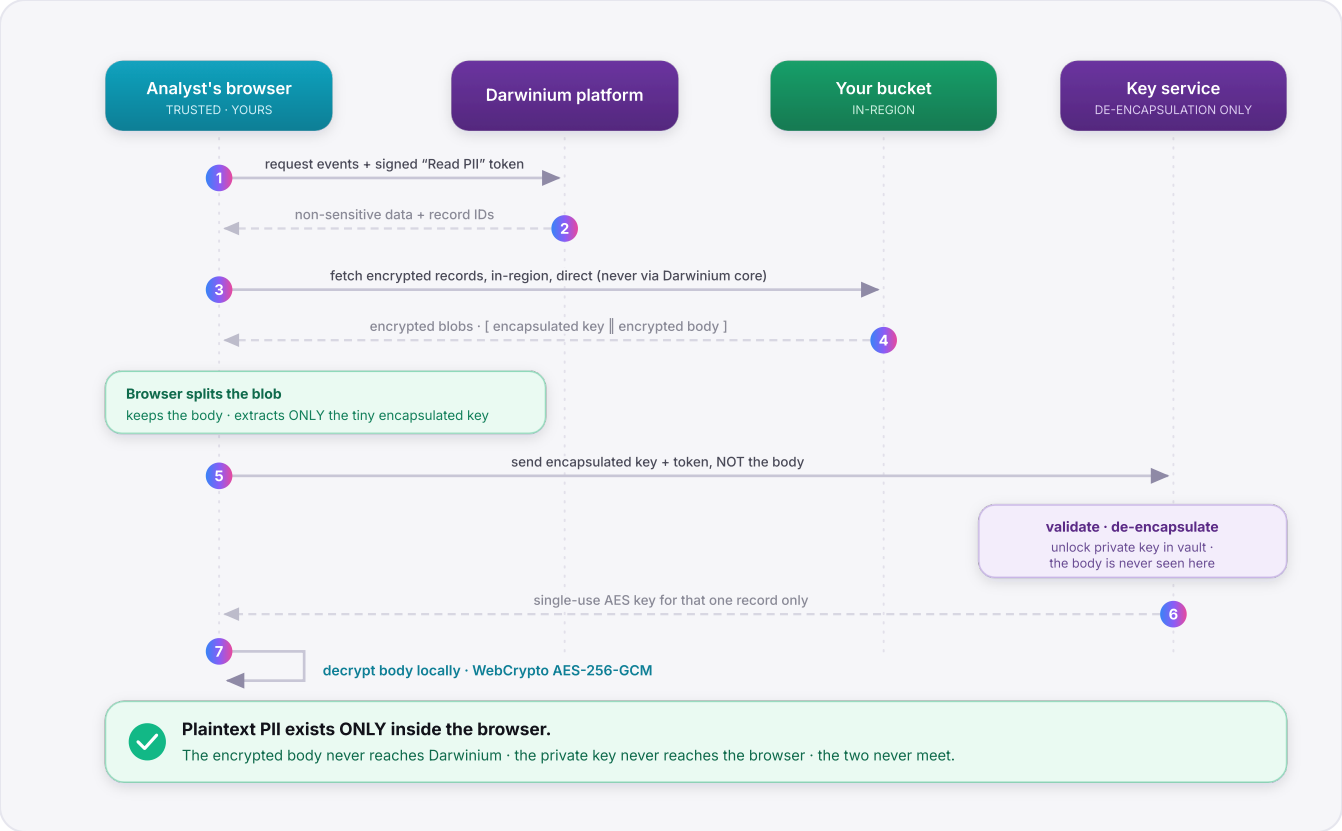


FIG 2 The browser only ever sends a few bytes of key material to Darwinium and gets back a single-use key. The encrypted body is decrypted inside the browser; plaintext never touches a Darwinium server, and the private key never reaches the browser.

The engineering behind it

The idea is simple, and rests on one detail of how HPKE is built. Every encrypted record is really **two pieces**: a small **encapsulated key** (the key-encapsulation output) and the **encrypted body** (the authenticated-encryption output). To recover plaintext you need the encapsulated key *and* the private key *and* the body, all in one place. **In-Browser Decryption deliberately never lets those meet.**

DARWINIUM'S HALF

Uses the private key to turn the *encapsulated key* into a single-use session key. **It never receives the encrypted body**, so it can never produce plaintext.

THE BROWSER'S HALF

Holds the *encrypted body* and receives the session key. **It never receives the private key**, only a disposable key for the one record on screen.

Split that way, no single party can reconstruct your customer's data, and the half Darwinium holds never includes the data at all:

WHO	ENCAPS. KEY	ENCRYPTED BODY	PRIVATE KEY	PLAINTEXT PII
Darwinium key service	sees	never	in-vault use	never
Your bucket	stores	stores	never	never
Analyst's browser	sees	sees	never	only here

Four more details make it production-grade, not just clever:

- **Fresh key per record.** Every event is sealed with its own single-use key, so records can't be correlated and one leaked key exposes exactly one record.
- **Latency hidden by design.** Keys and bodies are returned separately, so the browser starts de-encapsulating keys while bodies are still downloading; decryption overlaps the fetch.
- **Rotation is seamless.** A small digest travels with each record identifying which key sealed it, so rotating a key never breaks access to historical data.
- **No hand-rolled crypto.** The final decrypt uses the browser's own audited WebCrypto AES-256-GCM, not custom JavaScript cryptography.

Every step is gated by a signed permission token; the most sensitive categories (passport, PAN, biometric, health) require a separate claim and are available to humans only. And the guarantee holds *even though Darwinium runs the service:* because the two halves never meet, our servers derive a single-use key but **never receive the encrypted body, so plaintext PII exists only inside your analyst's browser, never on Darwinium infrastructure.**

02 THE API PATH

API Outpost, same privacy for server-to-server

Not every integration lives at the CDN edge. Many teams prefer to call Darwinium **server-to-server**. Done naïvely, that means sending PII to a vendor's API, exactly the exposure we designed out at the edge. **API Outpost** closes that gap.

API Outpost runs the Darwinium engine **inside your own AWS account**, deployed as a serverless function behind your own API gateway, and does everything the edge does: runs your rules, processes PII, encrypts it with HPKE, and writes the ciphertext **directly to your own bucket, with no part of that path touching Darwinium infrastructure.**

> *"Outpost allows Personally Identifiable Information to be processed and anonymized before it leaves your cloud environment... Darwinium is not able to access your customers' data... you control exactly in which region the data stays."*
>
> Darwinium documentation

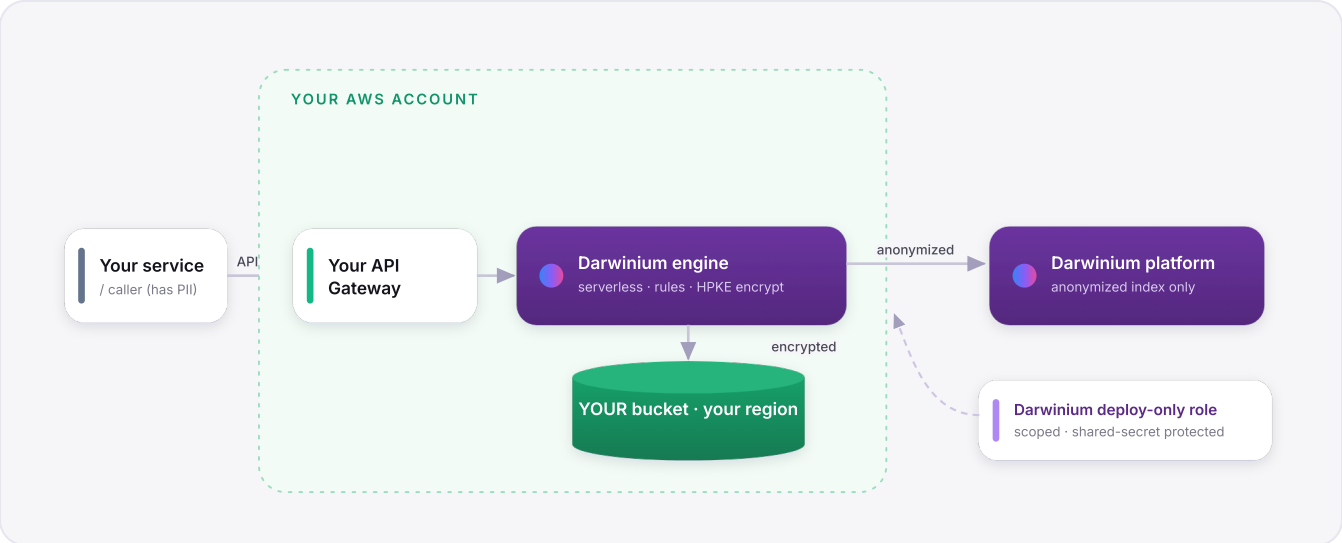


FIG 3 API Outpost runs the Darwinium engine inside your own AWS account. PII is processed and encrypted in your perimeter; only anonymized data leaves. Darwinium's only access is a narrow, deploy-only role.

HOW YOU INTEGRATE	WHERE PII IS ENCRYPTED	WHAT DARWINIUM RECEIVES
CDN edge (web / mobile)	Your CDN, inside your perimeter	Anonymized index only
API Outpost (server-to-server)	Your AWS account, inside your perimeter	Anonymized index only

However you connect, **PII is encrypted before it leaves your environment, and Darwinium never sees it in the clear, not even for a moment.**

03 KEY CUSTODY

Who can decrypt, and who provably cannot

An encryption architecture is only as strong as its key management. Darwinium's is built so that decryption requires a *specific human with permission*, and so that the automated processes attackers actually compromise structurally **cannot** decrypt.

- **A unique keypair per customer instance.** There is no master key; compromise is contained to a single instance.
- **Private keys live in a hardened secrets vault**, governed by policies that **prevent any automated process from ever reading them.**
- **A private key unlocks only for a signed token** carrying the **"Read PII" permission**, the single path to a key anywhere in the system.
- **Reading your bucket is human-only**, so a compromised Darwinium service cannot quietly drain it.
- **The most sensitive data is isolated further**: passport / PAN / biometric / health identifiers use a **separate keypair, path and policy**, humans-only, audit-logged separately.
- **Keys can be rotated** without breaking access to historical data; a digest on each record selects the right key.

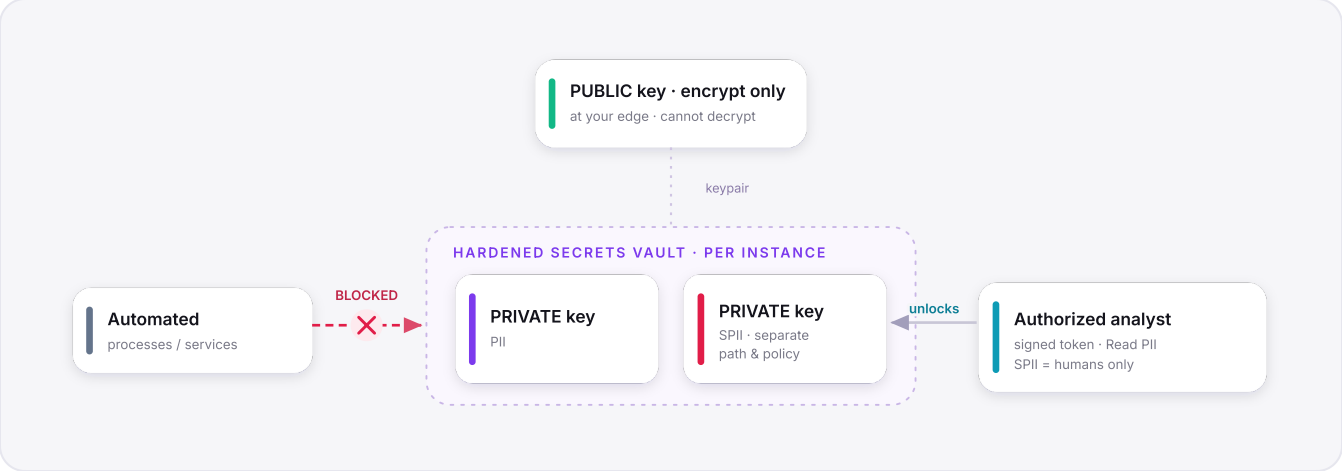


FIG 4 Encryption uses a public key at the edge; the private key unlocks only for a human with permission, never an automated process. The most sensitive data sits behind a second, isolated key.

THREATS ADDRESSED

Coverage that maps to your compliance register

This isn't privacy theater. It lines up, control for control, with the regulatory tests your legal and compliance teams already apply. *(Framing is general; applicability depends on your data, jurisdictions and configuration. Validate with counsel.)*

Cross-border transfer & data residency

GDPR · SCHREMS II

Schrems II[5] struck down Privacy Shield and now requires "supplementary measures" for transfers. The EDPB's own guidance (Recommendations 01/2020, "Use Case 1")[6] states strong encryption *is* an effective transfer safeguard, **provided the vendor never holds the keys.** Ciphertext in your region, keys you can hold yourself, is a textbook fit.

→ *Store in-region, keep the keys, shrink the transfer question to near zero.*

Security of processing & pseudonymization

GDPR ART. 32

Article 32[7] names encryption and pseudonymization as appropriate measures. PII encrypted with keys Darwinium cannot access, plus a one-way anonymized index, is unintelligible to us in practice. *(It strengthens your posture; it doesn't by itself change a processor's legal status.)*

→ *Demonstrable Art. 32 measures; minimized vendor handling of identifiable data.*

Breach-notification safe harbor

US STATE LAWS · HIPAA

Nearly every US state law includes an **encryption safe harbor**[9]: if stolen data was encrypted and the key wasn't also taken, notification generally isn't triggered. HIPAA treats properly encrypted health data as not an "unsecured breach"; HHS guidance[8] calls a provider that holds only encrypted data and no key a "no-view" service that "lacks access."

→ *A Darwinium breach yields keyless ciphertext, structurally inside the safe harbor.*

Third-party & operational-resilience risk

DORA · SOC 2 · ISO 27001

DORA[10] (in force January 2025) holds financial firms accountable for third-party ICT concentration risk. A vendor that never holds decryptable PII **materially shrinks the third-party attack surface** you govern, and maps cleanly onto SOC 2 Confidentiality criteria and ISO 27001 cryptography controls.

→ *Less concentrated PII at the vendor means less third-party risk to manage.*

Zero Trust / assume-breach alignment

NIST SP 800-207

Zero Trust is data-centric: protect the resource, assume the network is hostile. Encrypting data at the edge and decrypting it only inside an authorized human's browser is that principle made literal, "the provider can't read it," through key custody.

→ *An assume-breach posture where the assumption costs you nothing.*

Darwinium maintains ISO/IEC 27001:2022 and SOC 2 Type II, and, because it does not process cardholder data in the clear, has been attested out of scope for PCI DSS. Details at trust.darwinium.com.

WHY IT MATTERS

The blast-radius test

The clearest way to evaluate any data-privacy architecture is adversarial: *assume the vendor is fully breached tomorrow. What does the attacker walk away with?*

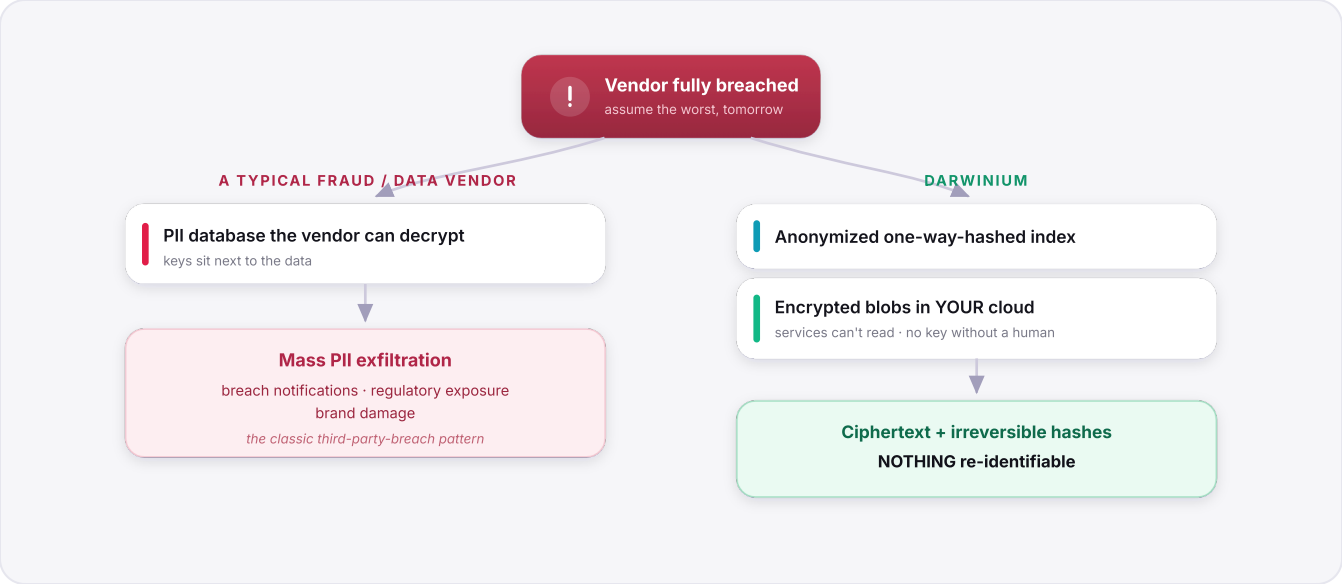


FIG 5 Same breach event, two radically different outcomes. A typical vendor loses readable PII; Darwinium loses ciphertext and irreversible hashes: nothing re-identifiable.

A TYPICAL FRAUD / DATA VENDOR

A database of customer PII the vendor's own systems can decrypt. One stolen credential or a server compromise → **mass PII exfiltration, breach notifications, regulatory exposure, brand damage.** The pattern behind every headline third-party breach.

DARWINIUM

An **anonymized, one-way-hashed index**, plus **encrypted blobs in your own cloud** that Darwinium's services can't read (human-gated) and can't decrypt (no key without a person). Attacker's take: **ciphertext and irreversible hashes. Nothing re-identifiable.**

This is why the word *over-engineered* is earned. Any one of these layers would be a strong story on its own; Darwinium stacks all four. Every other security control degrades under a determined enough attacker. **Data minimization is the one control that fails safe.**

Privacy as architecture, not policy

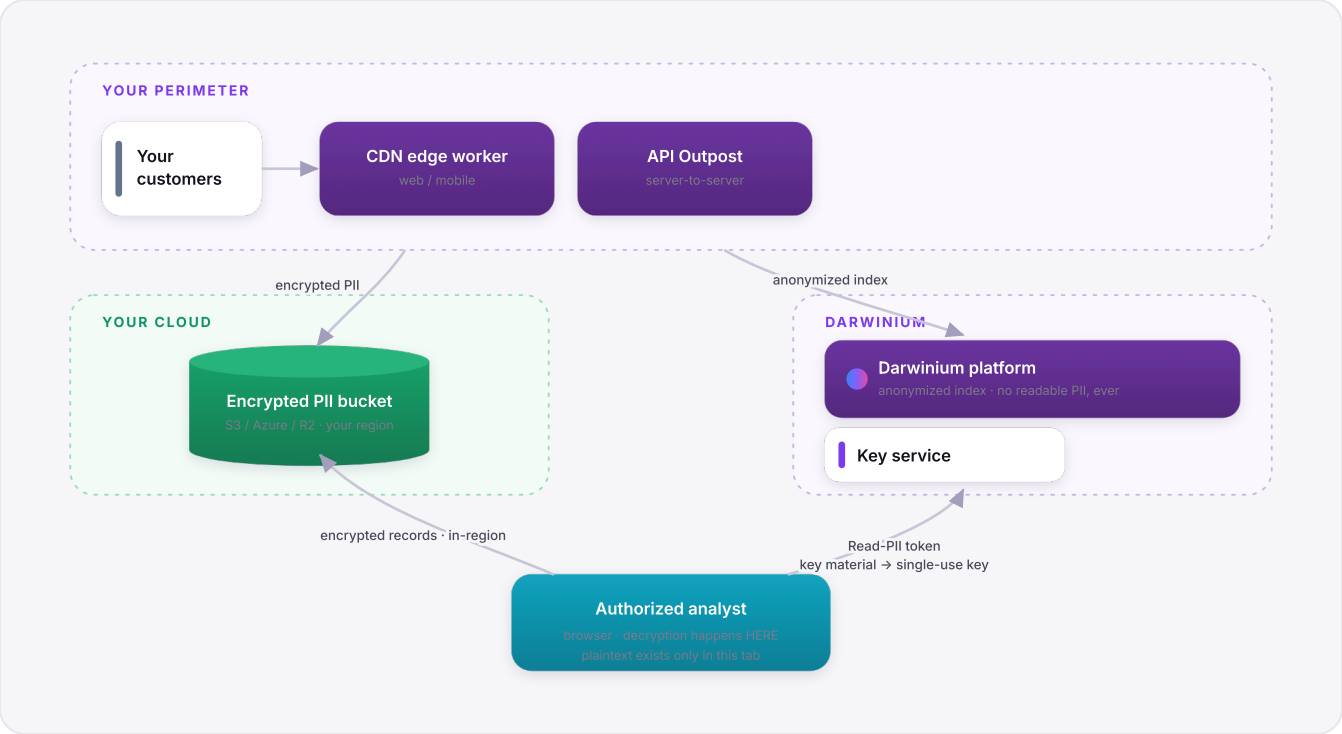


FIG 6 One deployment governs web, mobile and API. Encryption happens in your perimeter, encrypted data lives in your cloud, the anonymized index lives with Darwinium, and decryption happens only in the analyst's browser.

The convenience and scalability of the cloud, with the privacy of on-prem. You no longer have to choose between them.

- **Designed for breach, not against it.** Our posture doesn't rely on Darwinium being un-breachable; it relies on Darwinium holding nothing worth stealing.
- **You keep custody of the data.** Your bucket, your region, and the ability to read your own data yourself, without calling Darwinium at all. And because decryption happens only in your analyst's browser, we never see it in the clear.
- **One architecture, both ways in.** Edge or API Outpost, PII is encrypted inside your perimeter and Darwinium receives only anonymized data. No "just this once" exception.
- **Standards, not secrets.** HPKE (RFC 9180), AES-256-GCM, TLS 1.3, one-way elliptic-curve anonymization. All mainstream, auditable primitives, not a proprietary black box.
- **It maps to your obligations.** GDPR & Schrems II, HIPAA and US-state safe harbors, DORA, NIST Zero Trust.

Every vendor that holds your customers' PII is one stolen credential away from becoming the next headline. Darwinium chose not to hold it: the difference between *promising* to protect data and *engineering* so there's nothing to leak.

References

1. **IBM, Cost of a Data Breach 2025.** Average and US breach cost; mean time to identify and contain.
   https://www.ibm.com/reports/data-breach
2. **Verizon, Data Breach Investigations Report (DBIR).** Third-party share of breaches.
   https://www.verizon.com/business/resources/reports/dbir/
3. **NIST SP 800-207, Zero Trust Architecture.** "Assume breach" guidance.
   https://csrc.nist.gov/pubs/sp/800/207/final
4. **RFC 9180, Hybrid Public Key Encryption (HPKE).** IETF / IRTF CFRG.
   https://www.rfc-editor.org/info/rfc9180
5. **CJEU "Schrems II" (Case C-311/18).** Invalidation of Privacy Shield; transfer safeguards (overview via IAPP).
   https://iapp.org/news/a/the-schrems-ii-decision-eu-us-data-transfers-in-question
6. **EDPB Recommendations 01/2020.** Supplementary measures for transfers; "Use Case 1" (encryption with keys held only by the exporter).
   https://www.edpb.europa.eu/our-work-tools/our-documents/recommendations/recommendations-012020-measures-supplement-transfer_en
7. **GDPR Article 32.** Security of processing; encryption and pseudonymization.
   https://gdpr-info.eu/art-32-gdpr/
8. **HHS, HIPAA Breach Safe Harbor & Guidance (incl. FAQ 2076).** Encrypted ePHI; "no-view" cloud services.
   https://www.hhs.gov/hipaa/for-professionals/faq/2076/
9. **US state breach-notification laws (encryption safe harbor).** 50-state survey (Foley & Lardner).
   https://www.foley.com/insights/publications/2026/03/state-data-breach-notification-laws/
10. **DORA (Digital Operational Resilience Act).** Third-party ICT concentration risk (via ESMA).
    https://www.esma.europa.eu/esmas-activities/digital-finance-and-innovation/digital-operational-resilience-act-dora

Accessed 2026. Report figures update annually; confirm the latest editions before external publication.

Darwinium

ABOUT DARWINIUM

Continuous customer protection

Darwinium unites digital security with fraud prevention across every customer journey, protecting your end users from fraud, scams and online abuse, with full visibility of trust and risk across your entire digital estate. It works from **inside your own perimeter**: encrypting sensitive data at the edge, storing it in a cloud you control, and decrypting it only in the browser of an authorized analyst, so the platform that protects your customers never becomes a liability holding their data.

Continuous security analytics, customer-journey orchestration, and real-time action on evolving threats, through one integration, with a privacy architecture engineered so that a breach of the vendor leaks nothing.

Enable AI Commerce. Stop AI Fraud.

Darwinium: Continuous Customer Protection.