COMMERCIAL / TECHNICAL OVERVIEW

Data That Belongs to the Archer

How BowPro+ combines local-first ownership, purpose-specific trust and disciplined catalog governance into a resilient product architecture.

BowPro+ treats data management as part of the product, not as plumbing hidden behind the interface. Official catalogs, device recovery, user-created catalogs and person-to-person setup sharing are different operations with different risks. The architecture therefore gives each one its own trust model, package rules and import behavior rather than forcing every exchange through a universal file format or a permanent cloud account.

01 **Local-first ownership** Core equipment data, calculations, backups and sharing remain usable without an account or vendor server.	02 **Separated trust domains** Official releases, device recovery, user bundles and recipient shares do not depend on one master secret.
03 **Controlled, reversible imports** Packages are identified, authenticated, validated and previewed before a transactional commit.	04 **Privacy by architecture** Only selected data leaves the device; raw internal persistence and unrelated records stay local.

Data management is a product advantage

The practical value is broader than encryption. A resilient data layer lets the archer keep working during an outage, move equipment knowledge between devices and people, recover from mistakes, and trust that an official catalog is genuinely official. Those outcomes reduce support risk and make the BowPro+ ecosystem more useful over time.

Offline is not a degraded mode

- A range session, competition trip or remote hunt is not interrupted because a service is unreachable.
- The user owns the working copy of bows, shafts, vanes, sights, components and complete setups.
- Sharing can remain direct and deliberate: select the content, create a package, send it, inspect it and import it.
- Community expertise becomes portable without turning every useful workflow into an online dependency.

The central rule

BowPro+ data should remain useful when the network is unavailable, understandable when it moves, and recoverable when something goes wrong. Security, privacy and usability are strongest when they reinforce the same workflow.

Security without magical thinking

A client application cannot honestly claim that every embedded symmetric secret is impossible to recover. BowPro+ instead limits the value of any single compromise. Understanding a package format—or even extracting one application-level key—must not grant the ability to sign official releases, impersonate another installation or decrypt recipient-bound and device-bound packages.

BOWPRO+ TRUST ARCHITECTURE

Different Data, Different Rules

Four package classes, four explicit trust models—and no single secret whose loss collapses the whole system.

The package class determines who should be able to create it, who should be able to open it, what data behavior is permitted and what proof of origin is required. That separation is the foundation of a defensible offline architecture.

Package class	Protection and authority	Who can use it	Data behavior
Official catalog release	Authenticated encryption plus an official public-key release signature. The signing private key is never shipped in the app.	Any BowPro+ installation after signature verification.	Authenticated replacement of included system catalogs.
Device recovery package	Encrypted to and signed by persistent installation-owned keys.	Only the installation that created the backup.	Exact system-catalog recovery; a foreign installation is rejected.
User catalog bundle	Authenticated encrypted container plus exporting-installation signature.	Any BowPro+ installation selected by the user.	User-selected categories; append-only merge; duplicates skipped and reported.
Recipient setup share	Ephemeral key agreement, key derivation, authenticated encryption and sender signature.	Only the intended recipient installation.	Applies selected bow, arrow, sight or tape data without exposing raw internal persistence.

One master secret is not a security architecture

Legacy schemes often rely on an opaque transform and a static phrase embedded in the application. Once that phrase is recovered (TrustInGod, for example) and the data packet relies on trivial encoding scheme/XOR shifting, confidentiality, authenticity and tamper resistance can all disappear together. BowPro+ uses domain separation: official publishing authority, device-owned recovery keys, recipient agreement keys and package-specific encryption are independent boundaries.

A

Confidentiality

Package-specific keys limit who can read the payload.

B

Authenticity

Digital signatures identify who was authorized to create the package.

C

Tamper detection

Authenticated encryption and signatures reject modified headers, manifests and payloads.

D

Failure isolation

A compromise in one trust domain does not automatically unlock the others.

A modern, versioned package envelope

The secure container should be explicit rather than mysterious. A versioned envelope can identify the package class, schema version, content type, algorithms, sender or recipient context, nonce, authenticated manifest and payload. This supports controlled evolution, migration and deprecation without treating obscurity as the security mechanism.

A claim BowPro+ can defend

Reverse engineering the format does not create publishing authority. A person who understands the bytes still cannot manufacture a valid official release without the private signing key, or open a package bound to another installation without the corresponding private key.

Architecture note: algorithm identifiers and schema versions should remain explicit so cryptographic and data-format migrations can be introduced without breaking every existing package at once.

BOWPRO+ DATA GOVERNANCE

Import Without Surprises

The safest import is one the user can understand before it changes anything—and one that either completes fully or changes nothing.

Cryptography proves origin and protects content, but it does not decide whether a record is valid, compatible or desirable. BowPro+ therefore treats import as a staged transaction with visible policy. The package is inspected in memory, validated against the active schema and catalog rules, summarized for the user, then committed atomically.

1. Identify the package

Read the outer envelope, classify the package and reject unsupported versions, content types, sizes or malformed structures before processing the payload.

2. Authenticate and decrypt

Verify the required signature and authenticated-encryption metadata before trusting the contents. Failure is closed and explicit.

3. Validate the data model

Check schema, identifiers, required fields, units, component compatibility and category-specific constraints. A valid signature does not excuse invalid data.

4. Preview the effect

Show what will be added, replaced, skipped or rejected. The user chooses the categories and confirms the change instead of importing a black box.

5. Commit atomically—or not at all

Create the appropriate local backup, write all selected changes as one transaction and roll back on any failure. Partial imports are not accepted as success.

Different workflows keep different promises

Workflow	User-facing rule	Technical discipline
Official update	“This is a genuine BowPro+ release and these catalogs will be replaced.”	Verify authority first; validate checksums and schema; replace atomically.
Device recovery	“This package belongs to this installation and restores its exact protected state.”	Reject foreign-device backups; create a local rollback copy before restore.
User catalog import	“Your existing records win; only selected new records are appended.”	Preflight duplicates; leave unselected categories untouched; report added and skipped items.
Recipient setup share	“This package was intended for this installation and contains only the shared setup data.”	Recipient-bound decryption; validate portable schema; avoid exposing internal database representation.

Duplicate handling is governance, not guesswork

The import policy needs stable, explainable identities. Typical examples include normalized brand + model + axle-to-axle length for bows; brand + model + spine for shafts; brand + model + vane length for vanes; and manufacturer/name/type plus explicit match-rule identifiers for components. The exact key can evolve, but it must be deterministic, versioned and visible in diagnostics. Good recovery is planned before the write begins. BowPro+ can preserve the current state, reject a package that belongs to another trust domain, restore exact protected catalogs where appropriate and explain the result. Before confirming an import, the user should be able to answer four questions: who created it, who it was intended for, what will change and whether recovery is possible. This turns catalog maintenance from emergency file surgery into a predictable workflow.

BOWPRO+ PRIVACY AND SECURITY

Protect Less Data, Protect It Better

Privacy is not only stronger encryption. It is data minimization, local ownership, narrow package scope and clear user intent.

Privacy is also what the system does not require

01

No BowPro+ account

Core data ownership, backup and package workflows do not require a central identity service.

02

No mandatory server path

A package can be created, transferred, verified and imported without a BowPro+ cloud round-trip.

03

No silent full-data export

The user selects the categories and setup elements included; unrelated local records remain on the device.

04

No raw database sharing

Portable, versioned package schemas expose the intended data—not the application's internal persistence layout.

Threats and the corresponding control

Threat or failure	BowPro+ response
Modified or truncated package	Authenticated encryption, signatures and explicit manifest validation reject the package before commit.
Forged official catalog update	Only the offline-controlled release-signing private key can authorize an official release; the app holds verification material only.
Package forwarded to the wrong person	Recipient-bound encryption prevents another installation from opening the share.
Reverse-engineered application secret	Separated trust domains prevent one recovered client secret from granting official authority or other installations' keys.
Malicious, oversized or incompatible import	Size limits, bounded parsing, strict schemas and transactional writes contain the failure.
Sensitive traces in diagnostics or temporary files	Logs exclude payloads and secrets; decrypted buffers and temporary artifacts remain minimal and short-lived.
Device loss or replacement	User-controlled export and backup provide continuity without a central BowPro+ copy of the user's equipment data.

Recommended production discipline

- Use platform-protected key storage and non-exportable keys where practical.
- Use reviewed cryptographic libraries and standard constructions—never custom cryptography.
- Guarantee unique nonces and keep algorithm, key and schema versions explicit.
- Authenticate before commit; validate untrusted lengths and test corrupted, replayed and cross-device packages.
- Keep metadata minimal, exclude secrets from logs and expose optional notes before export.
- Provide key-rotation and format-migration paths so secure data remains readable tomorrow.

Wrap-up

BowPro+ combines offline availability, local ownership, modern cryptographic separation and disciplined data-governance rules. It is not security theatre built around a hidden phrase; it is a recoverable, auditable and extensible package system designed around real archery workflows. The result is safer publishing, controlled collaboration, lower recovery risk and continuity without turning data ownership into a subscription to server availability.

Modernized architecture overview • August 2026