XFactor: Multi-Step, Phishing-Resistant Step-Up Across Every Channel

Trust & Risk / Last updated 2026-06-11 / https://www.scrambleid.com/learn/scrambleid-xfactor-step-up

> **Status (June 2026):** In development. XFactor is on the ScrambleID roadmap and isn't shipping yet. This article describes the design we're building toward, published now so architects evaluating the model can plan around it. If a near-term deployment depends on XFactor behavior, talk to your ScrambleID account team about current timelines before committing.

In one sentence: XFactor runs a policy-selected chain of **phishing-resistant** checks before allowing a sensitive action, and returns a **signed result** that your app (web, IVR, people verification, desktop, M2M) can verify and audit.

TL;DR (canonical)

- Step-up is not "more prompts"; it is **higher assurance bound to intent**.
- A chain is phishing-resistant when it uses **public-key proof**, **origin/session binding**, and **no reusable secrets**.
- XFactor is intentionally opinionated: **no SMS/voice/WhatsApp OTP factors** for high-risk policies. SMS and voice OTPs are defeated by SIM-swap attacks, real-time relay through adversary-in-the-middle phishing kits, and direct social engineering of cellular carriers; WhatsApp OTPs add account-takeover and number-porting risk on top of that. Phishing-resistant chains start from the assumption that any OTP a user can read aloud or copy can be intercepted.
- Publish your step-up KPIs (rates, pass/fail, abandon, median time) so the program is measurable and citeable.

Why step-up exists

Most account takeovers are not defeated at the first login prompt. Attackers often target **high-blast-radius actions**:

- password resets and recovery
- payout destination changes
- new device enrollment

- admin settings changes (SAML keys, redirect URIs)
- data exports

If you force high-assurance checks on every low-risk moment, you lose adoption. If you never step up, you lose money moments.

XFactor gives you the middle path: **fast baseline + strong gates when it matters**.

Factor catalog (designed-in)

XFactor is designed to compose factors across channels. Common designed-in factors:

Factor	What it proves	Channel fit	Notes
WebAuthn (User Verification, UV: device biometric or PIN unlock)	user presence + local UV + origin binding	web, desktop	use UV-required for sensitive actions
Signed QR (DID) + Type Code	cross-device intent binding	web, desktop, IVR	typed code prevents "approve-from-anywhere"
Hardware security key	strong phishing resistance	web, desktop	excellent for privileged workforce
Device re-check (biometric/PIN)	that the device holder is present	mobile/desktop	use as a gate before revealing sensitive attributes
Network / IP policy check	environmental constraint	web, M2M	never use alone; treat as supplemental

---

Chain examples (recommended)

Workforce privileged action

- WebAuthn (UV required) → Hardware key

Consumer high-value change

- Signed QR (DID) + Type Code → WebAuthn (UV)

Contact center reset (mid-call)

- Caller DID confirmation → WebAuthn (UV) or device re-check

M2M token minting with human approval

- Human custodian WebAuthn approval → mint short-lived scoped token

How XFactor fits ScrambleID

> **Binding for QR(DID) and WebAuthn factors:** the cryptographic specification of how each factor binds to the initiating session and intent, including the threat model coverage and what attacks are defeated by binding versus what attacks are not, is in Session binding cryptography.

1. Your surface detects a risky action.
2. It requests a chain run: `requireXFactor(chainId, context)`.
3. XFactor orchestrator binds the chain to the initiating session and streams state.
4. On completion, XFactor returns a signed result artifact (XFR).
5. Your surface allows/denies and writes audit.

Policy object (concept)

```
{
  "chainId": "CHAIN_WORKFORCE_PRIV",
  "bindTo": {
    "sessionId": "sess_...",
    "origin": "https://app.example.com",
    "action": "change_saml_certificate"
  },
  "ttlSeconds": 120,
  "denyOnTimeout": true
}
```

UX patterns that prevent social engineering

- Always show **what action** is being protected.
- Show **who is requesting** (app name, domain, call center agent system).
- Use a typed code when approving cross-device.
- Make timeouts explicit.

Accessibility + localization (make strong step-up usable)

Phishing-resistant step-up only works if real users can complete it.

Accessibility checklist

- [ ] **Time limits are understandable.** If a request expires, tell the user and offer a simple retry.
- [ ] **Don't rely on color alone.** Required steps and status should be announced with text/icons.
- [ ] **Screen-reader friendly.** Read the protected action first, then the steps, then the confirm button.
- [ ] **Avoid tiny numeric entry fields.** Provide large inputs for typed codes and support paste where safe.
- [ ] **Voice alternative when appropriate.** For call flows, pair the DID with a simple DTMF option.

Localization checklist

- Keep policy identifiers stable (e.g., `CHAIN_CONSUMER_PAYOUT`); localize only display strings.
- Use locale-aware number reading for DIDs (digit-by-digit).
- Make the protected action unambiguous in every language (avoid idioms).

Copy templates (what the user sees)

- Header: **"Confirm to change payout destination"**
- Subheader: "This request came from **{AppName}** on **{Domain}**."
- Code hint: "Enter the 6-digit code shown on your screen."
- Failure: "Verification failed. Nothing changed. Try again or contact support."

Anti-patterns (do not ship)

- "Approve from anywhere" push prompts.
- weak fallbacks (SMS OTP, email OTP) on high-risk actions.
- long chains that feel like punishment.
- unclear error states that generate helpdesk tickets.

Rollout plan

1. Start with the top 2-3 highest-risk actions.
2. Instrument chain invocation and outcomes.
3. Tune: abandon rate, step durations, failure reasons.
4. Expand to more actions once stable.

What to measure

- step-up rate (by action)
- pass rate / fail rate / timeout rate
- median and p95 time to complete chain
- abandon rate
- attack deflection signals (declined approvals, wrong-code spikes)

(See: Metrics + ROI Playbook)

Key Takeaway

XFactor is ScrambleID's multi-step step-up orchestrator that runs a policy-selected chain of phishing-resistant checks before allowing sensitive actions. Unlike single-factor step-up, XFactor can require multiple sequential proofs (biometric + hardware key + manager approval) based on action risk. All step-up attempts are logged with correlation IDs for audit.

FAQ

When should we trigger step-up?

When an action is high-risk (payout change, password reset, admin settings) or when risk signals indicate an active attack.

What makes a chain phishing-resistant?

Public-key proof + origin/session binding + no reusable secrets.

Can we run XFactor in voice flows?

Yes. Caller flows can gate call outcomes on XFactor completion.

Is SMS OTP supported as a step-up?

Not for high-risk policies. OTPs are replayable and frequently defeated by social engineering.

How do we prevent fatigue?

Keep chains short and targeted. A strong two-step chain beats a long checklist.

References (public)

- WebAuthn (W3C): https://www.w3.org/TR/webauthn/
- CISA phishing-resistant MFA: https://www.cisa.gov/sites/default/files/publications/fact-sheet-implementing-phishing-resistant-mfa-508c.pdf
- NIST Digital Identity Guidelines (general assurance concepts): https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-4.pdf
- WCAG 2.2 (timeouts / enough time): https://www.w3.org/TR/WCAG22/

Related reading

- Overwatch: Risk Engine
- Lockstep: Dual Control
- Caller Authentication