Caller Authentication: Replace KBA and Stop Vishing (Phone-Channel Verification)

Voice & Contact Center / Last updated 2026-06-11 / https://www.scrambleid.com/learn/caller-authentication-stop-vishing

In one sentence: ScrambleID Voice replaces phone-channel KBA with cryptographic device proof: when the caller's number matches a registered device, a **push confirmation** in the ScrambleID app verifies them with one tap, and when it doesn't, a **session-bound Dynamic Identifier (DID)** is read on the call and confirmed in the app, so attackers can't succeed with persuasion, spoofed caller ID, or deepfake voice.

TL;DR (canonical)

- Caller ID (ANI) is **context**, not identity. It can be spoofed.
- KBA fails because answers are static and replayable, NIST SP 800-63A-4 restricts its use; vishers weaponize pressure and OSINT.
- The fast path: the caller's number matches a registered device, a push confirmation lands in the ScrambleID app, and one tap (optionally gated by a local biometric or PIN) verifies the caller.
- The fallback: a short-lived, single-use DID spoken in the IVR (or by an agent), then confirmed cryptographically in the app.
- When confirmed, the IVR call flow updates in real time, fast enough that the caller never notices a wait.
- You get deterministic audit trails and measurable indicators (wrong-code spikes, time-to-verify, containment).

What problem does Caller Auth solve?

Contact centers are the most common place a modern account takeover gets *completed*.

Why? Because the phone channel combines:

- **high urgency** ("my account is locked"),
- **human judgment** (agents making exceptions),
- **weak legacy controls** (KBA, SMS fallback), a gap highlighted by CISA,
- and **spoofable signals** (caller ID, voice similarity).

Caller Auth is designed to eliminate the two weakest links:

1. **knowledge checks spoken over the phone**
2. **agent-based judgment** as the main gate

How does Caller Auth work?

> **Cryptographic detail:** the signed confirmation payload structure, what "device-bound keys" means at the OS level (Secure Enclave, TEE, TPM), and how the verifier matches the call session id atomically are specified in Session binding cryptography and Device key lifecycle in the architecture reference.

The flow starts from the caller's number. ANI is never trusted as identity (it can be spoofed), but it's a useful lookup key.

The fast path (most enrolled callers):

1. The call connects and the IVR creates a verification session.
2. The caller's number matches a device registered to the account, so ScrambleID sends a push confirmation to the ScrambleID app on that device.
3. The caller taps to approve, optionally unlocked with a biometric or PIN locally on the device. The app signs the challenge with device-bound keys.
4. The IVR receives a callback and moves the live call to a verified state.

The fallback (unrecognized number, or push unavailable):

1. The IVR creates a verification session.
2. ScrambleID issues a short numeric DID with a TTL.
3. The IVR speaks the DID to the caller.
4. The caller enters the DID in the ScrambleID app.
5. The app signs and confirms the DID using device-bound keys.
6. The IVR receives a callback and moves the live call to a verified state.

Both paths end the same way: a cryptographic confirmation from a registered device, bound to the live call session. The spoken code is the honest fallback, not a weaker second system.

Why this is phishing-resistant for voice

- The push challenge and the DID are **single-use** and **short-lived**.
- Either is only meaningful when **confirmed by the app**.
- The confirmation is **bound to the IVR call session**.

So a vishing attacker who convinces an agent can't win without the customer's bound device.

What are the Caller Auth flow states?

State	What it means	What the agent/IVR should do
`pending`	DID issued; awaiting app confirmation	Wait loop; prompt for confirmation
`confirmed`	DID confirmed by app (cryptographic proof)	Route to success handler; allow sensitive workflow
`wrong_code`	Caller typed wrong DID in app	Continue waiting; retry; alert on bursts
`timeout`	DID expired without confirmation	Fail gracefully; optionally re-issue
`canceled`	Caller canceled verification (DTMF)	End verification; route accordingly

What integration options exist for Caller Auth?

Both options document the spoken-code webhook flow, which is the customer-side integration surface. The push fast path is part of the caller experience inside the ScrambleID app; it doesn't require a separate integration.

Option A, Automated IVR verification (recommended)

- IVR requests DID session via webhook.
- IVR speaks DID.
- IVR enters a wait loop (e.g., "Press 1 when complete").
- ScrambleID confirms out-of-band and redirects the live call.

Option B, Agent-assist verification

- Agent triggers verification from console (or IVR routes to agent first).
- Agent reads DID.
- Caller confirms in app.
- Agent UI updates to "Verified."

What should your Caller Auth checklist include?

- [ ] Identify KBA surfaces (password reset, payout change, device recovery).
- [ ] Choose one pilot queue with measurable volume.
- [ ] Define policies: DID length, TTL, retries, locale prompts.
- [ ] Integrate webhooks and verify provider signatures.
- [ ] Update IVR prompts and agent scripts.
- [ ] Instrument metrics (containment, time-to-verify, wrong-code rate).

- [ ] Remove KBA fallback (or gate exceptions with dual control; Lockstep, in development, is designed for this).

What scripts reduce social engineering risk?

IVR script (minimal)

> "To verify your identity, open the ScrambleID app. Your one-time code is 1-2-3-4-5-6. Enter it in the app. Do not say it out loud."

Agent script (minimal)

> "I'm going to verify you using the ScrambleID app. I will read you a one-time code. Please enter it in the app, don't say it out loud. Once it's approved, we can proceed."

Objection handling: "Just text me the code"

> "No, that would be less secure. The code only works inside ScrambleID. If anyone asks you to share it by text or email, it's a scam."

What metrics should you track for Caller Auth?

Start with these (definitions in Metrics + ROI Playbook):

- **caller containment %**
- **time-to-verify (median and p95)**
- **wrong-DID rate**
- **retry pressure** (number of loop prompts)
- **late confirmations** (confirmation after hangup)

How does Caller Auth relate to STIR/SHAKEN?

STIR/SHAKEN is a network-layer caller ID authentication framework that helps carriers assess whether a call's number is spoofed.

A practical thing to understand about STIR/SHAKEN: it produces an **attestation level** the receiving carrier can read, not a binary "real or fake" answer.

- **Attestation A (Full Attestation):** the originating carrier authenticated the caller and confirmed the right to use the calling number. Highest signal.
- **Attestation B (Partial Attestation):** the originating carrier authenticated the caller but did not verify the right to use the number. Mid signal; common for some enterprise PBX scenarios.
- **Attestation C (Gateway Attestation):** the call entered the network through a gateway and the originating carrier could not authenticate the caller. Low signal.

A call without any attestation, or with attestation C, is more likely to be spoofed but is not necessarily malicious. A call with attestation A is more trustworthy but is not necessarily safe: STIR/SHAKEN authenticates the *number*, not the *person*. A legitimate number can be used by anyone with the device.

Caller Auth complements STIR/SHAKEN at a different layer:

- STIR/SHAKEN labels whether the number is asserted/authenticated by the network and at what attestation level.
- Caller Auth verifies whether the caller can complete a cryptographic, session-bound challenge tied to a specific enrolled device.

That means Caller Auth remains effective even when caller ID is spoofed, when attestation is weak (B or C), or when brand labeling is ambiguous. It also adds value on top of strong attestation (A) by verifying the person rather than the number.

Key Takeaway

Caller authentication replaces knowledge-based authentication (KBA) with cryptographic proof. When the caller's number matches a registered device, a push confirmation in the bound mobile app verifies them with one tap; otherwise the IVR speaks a short-lived Dynamic Identifier (DID) that the caller confirms in the same app. This stops vishing because attackers cannot succeed with persuasion, spoofed caller ID, or deepfake voice, they need the customer's physical device to produce a valid signature.

FAQ

What is vishing?

Vishing (voice phishing) is social engineering conducted over the phone. Attackers impersonate legitimate organizations, banks, tech support, government agencies, to trick victims into revealing sensitive information, authorizing transactions, or installing malware. Vishing attacks have increased significantly as web security has improved, making the phone channel the path of least resistance for fraud.

What is caller authentication?

Caller authentication is a verification method that confirms a caller's identity without relying on knowledge-based authentication (KBA) like security questions or account details. Modern caller authentication uses cryptographic proof: a push confirmation to the caller's registered device, or a short-lived identifier presented on the call and confirmed through the same device-bound mobile app, proving the caller possesses the registered device.

How does caller authentication prevent deepfake voice attacks?

Deepfake voice attacks synthesize a victim's voice to fool agents or voice biometrics. Caller authentication defeats deepfakes because verification happens through a separate device channel, the caller must confirm a session-bound code in their authenticated mobile app. The attacker would need the victim's physical device, not just their voice.

What's the difference between Caller Auth and an OTP read on the phone?

An OTP read over the phone is still a replayable secret, if an attacker is on the line, they hear it. A Dynamic Identifier (DID) is confirmed in the app with device-bound keys and is bound to a specific call session. The DID itself is not a secret; it's a session artifact that requires cryptographic proof to complete.

Does Caller Auth require customers to speak sensitive information?

No. Nothing sensitive is spoken on the call. The DID is not a secret to be protected; it is a one-time session artifact that requires app confirmation.

What happens if the caller types the wrong DID?

The app rejects the confirmation; the session remains pending until timeout or cancellation. Wrong-code bursts can be an attack signal.

What about customers without smartphones?

Provide an assisted recovery option that is slower, audited, and harder to exploit, and avoid reintroducing KBA as the easy bypass.

Can agents bypass Caller Auth?

Your program should treat verification as system-driven. If exceptions exist, they should be audited and ideally require co-approval (Lockstep's job once it ships).

How fast is the experience?

Fast enough to beat KBA by an order of magnitude. The push fast path is a single tap, and even the spoken-code fallback takes a fraction of a KBA interview. The operational target is fast DID issuance

and fast post-confirm callback.

References (public)

- FCC overview of caller ID authentication and STIR/SHAKEN: https://www.fcc.gov/call-authentication
- CISA guidance on social engineering: https://www.cisa.gov/news-events/news/avoiding-social-engineering-and-phishing-attacks
- NIST SP 800-63A-4 (constraints on KBV/KBA approaches): https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63A-4.pdf

Related reading

- IVR Integration Guide
- KBA Is Dead: Contact Center Playbook
- Circle of Trust (trust cues)