ScreenBlur — Security & Anti-Tamper Model

This document is the honest threat model for ScreenBlur: what actually stops people from ripping us off, what only raises the bar, and — just as important — the tricks we deliberately **did not** use because they would make the app look like malware and fail corporate IT review.

TL;DR

- **The real lock is server-side, not client-side.** Enforcement lives in Ed25519-signed tokens minted by the license server. A cracked or patched client is worthless because it still cannot forge a valid signature.
- **The client is hardened, not "protected."** We compile to native code (Nuitka) and **Authenticode code-sign** the binary. That is the honest, AV-friendly ceiling for a desktop app.
- **We avoided the "obvious" anti-tamper tricks on purpose** — UPX packing, PyArmor obfuscation, anti-debugger traps. They trip antivirus heuristics and are the #1 reason a legit app gets quarantined. Passing IT review is worth more than cosmetic obfuscation.

---

1. The threat model, stated plainly

There is one unavoidable truth in desktop software: **any code that runs on a machine you don't control can eventually be reverse-engineered.** Anyone promising "uncrackable" client-side protection is selling snake oil. So we do not bet the business on hiding logic in the client. We bet it on cryptography the client cannot fake.

Attacker goal	What stops them
Read/modify the Python/source	Native compilation (Nuitka) — no `.py` / `.pyc` to unzip. Raises effort from “unzip the exe” to “reverse machine code.”
Patch the binary to skip the license check	Authenticode signature breaks on any edit → Windows/SmartScreen/IT flags it as tampered. And even a patched client still can’t mint a valid token.
Forge or extend a license / trial	**Impossible without the server’s private key.** All grants are Ed25519-signed; the client only holds the public key and verifies.
Share one key across many machines	Server hardware-locks a key to the first device’s irreversible fingerprint; second device is rejected.
Farm unlimited free trials	Trial count is tracked **server-side per device fingerprint** (default 3), so deleting local files does not reset it.
Roll the clock back to keep a trial alive	Trial end time is the server-signed token `exp`, verified against the signature — not local wall-clock the user can change.

---

2. What does the real work: server-side crypto

- The license server holds an **Ed25519 private key** (`signing_private_key.pem` , never shipped, never in git).
- Every grant it issues — a license validation token **and** a trial token — is a small JSON payload **signed** with that private key.
- The client (`licensing.py`) embeds only the **public key** (`PUBLIC_KEY_PEM`) and verifies signatures. It can read a token; it can never create one.
- Result: the interesting logic (“is this allowed, and until when?”) is decided by data that is cryptographically unforgeable. Cracking the client UI gets an attacker nothing they can’t already do by just… not running the app.

This is the part that must stay correct. For production, regenerate the key pair with `generate_keys.py` and paste the new public key into `licensing.py` before building — so no one else ships a client that trusts our demo key.

---

3. Client hardening (raising the bar, AV-friendly)

3.1 Native compilation — Nuitka (`build_hardened.bat`)

Instead of PyInstaller (which just bundles your `.pyc` files in a zip anyone can extract), the hardened build uses **Nuitka** to compile the Python to real C / native machine code. There is no source or bytecode sitting inside the exe to lift out. This is the single biggest client-side effort increase, and unlike obfuscators it produces a clean, ordinary native binary that antivirus trusts.

```
build_hardened.bat
```

> **Note on the console flag.** The script uses `--windows-console-mode=disable` (correct for current Nuitka). A few Nuitka versions have a bug where `disable` crashes PyQt5 GUI apps on launch — if the built exe won't start, change that flag to `--windows-console-mode=attach` and rebuild.

3.2 Authenticode code signing — do this, it matters most

A code-signing certificate is the highest-leverage anti-tamper step for a Windows app that must pass IT review:

- Any modification to the exe **invalidates the signature** — tamper-evident by design.
- Windows SmartScreen and corporate allow-lists trust signed binaries; unsigned ones get scary warnings or are blocked outright.

Sign the built exe with an OV/EV code-signing cert:

```
signtool sign /fd SHA256 /tr http://timestamp.digicert.com /td SHA256 ^
  /a dist\ScreenBlur.exe
```

(An **EV** certificate additionally gives instant SmartScreen reputation.)

3.3 UPX turned OFF

`screen_blur.spec` sets `upx=False`. UPX-packed executables are one of the strongest single signals antivirus engines use to flag malware. Smaller exe is not worth a quarantine.

---

4. What we deliberately did NOT do, and why

Tempting trick	Why we skipped it
UPX / other exe packers	Classic malware signature → AV false positives.
PyArmor / heavy bytecode obfuscation	Runtime unpacking stubs look like malware; breaks under some AV; only delays a determined RE by hours.
Anti-debugger / anti-VM traps	Behaves exactly like malware evasion; gets the app quarantined and fails security review.
Silent self-reinstall / watchdog processes / startup persistence	The definition of unwanted software. ScreenBlur installs no services, no drivers, no startup entries.

Our north star: **a corporate security team should be able to inspect this app and find nothing suspicious.** Obfuscation that fights the analyst also fights the IT reviewer we need on our side.

---

5. Trial mode abuse-resistance (summary)

- Default trial: **120 s** (`LICENSE_TRIAL_SECONDS`), **3 trials per device** (`MAX_TRIALS_PER_DEVICE`) — both server env-configurable.
- The app requests a trial token from `POST /api/trial`. When the device's server-side count is exhausted the endpoint returns **429** and no token.
- The countdown end is the **signed token** `exp`, so local clock changes, file deletion, or reinstalls don't grant extra or longer trials.
- When the trial expires the overlay **disables the acrylic blur** (`disable_acrylic`, screen becomes fully visible) and prompts for a key.

---

6. Where to keep looking (defense in depth, later)

- Rotate the signing key if the private key is ever exposed; the public key in the client pins trust to a specific server.
- Consider short token lifetimes + the existing offline grace window to balance UX against a stolen-token replay.
- Server is the crown jewel: keep `signing_private_key.pem` and the DB on a locked-down, persistent host; never commit either.