TINYWATCH SECURITY OVERVIEW

# A private monitor that never leaves your home.

TinyWatch turns two phones into a baby monitor that streams locally over your Wi-Fi — no cloud, no account, no third-party servers carrying your video. This document explains, in plain language, exactly how we make that promise hold.

### Local only

Video and audio travel directly between the two phones over your Wi-Fi or Personal Hotspot. They never touch our servers.

### Encrypted in flight

The video and audio stream uses WebRTC's mandatory DTLS-SRTP encryption — the same standard used by major video-call apps.

### One-time pairing

Each Camera Phone has a 48-bit secret embedded in its QR code. The Monitor must know that secret to start streaming.

The rest of this document walks through that promise in detail: how the pairing works, what encryption is in place, how we slow down brute-force attacks, and the small handful of things TinyWatch deliberately *doesn't* protect against.

01 · ARCHITECTURE

# How a TinyWatch session is set up

Every TinyWatch session is a direct conversation between two phones on the same local network. Nothing in the loop runs on our servers.

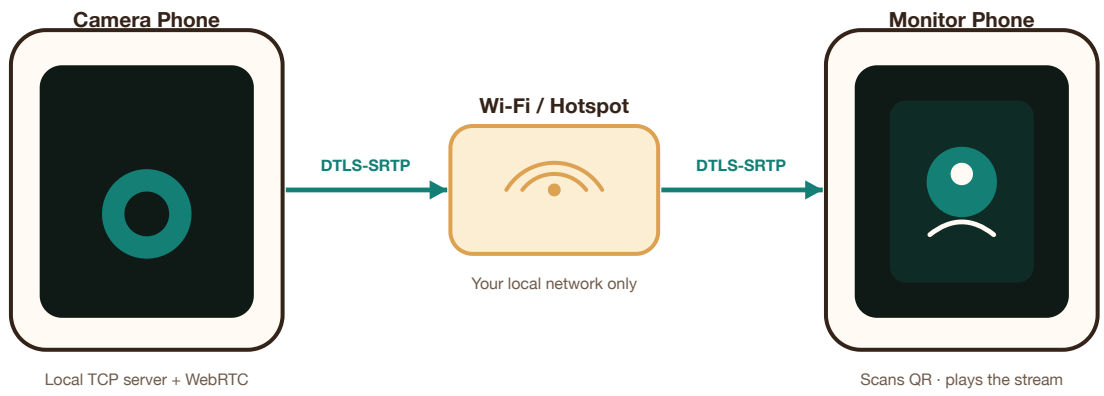


Figure 1 · The session has three real parties: two phones and the Wi-Fi network in between. There is no fourth party.

## What happens in order

- **The Camera Phone** starts a small local TCP server (port 58212 by default) and shows a QR code that contains its private IP address, the port, and a fresh random pairing secret.
- **The Monitor Phone** scans the QR code, reads the secret, and asks the Camera Phone for a WebRTC offer authenticated with that secret.
- **WebRTC takes over.** Video and audio flow peer-to-peer between the two phones, carried inside DTLS-encrypted SRTP packets — exactly the same transport modern video-call apps use.

### WHAT GOES THROUGH OUR SERVERS

Nothing. The TinyWatch app does not relay your video, your audio, or your phones' local IP addresses to any TinyWatch infrastructure. Everything stays on the two phones and the Wi-Fi (or Personal Hotspot) between them.

02 · PAIRING

# Who is allowed to connect

Anyone on your local network can probably *see* that TinyWatch is running. Only someone who has the pairing secret can actually start a stream.

## The secret in the QR code

Every time Camera Mode starts it generates a fresh random secret using your operating system's cryptographic random number generator (the same source modern apps use to generate passwords). The secret is **48 bits** long, encoded as a short URL-safe string in the QR code.

The QR code also carries the camera's local IP and port. None of these values mean anything outside your home network.

**48**
bits of entropy per secret

**90**
days the secret is valid

**50**
wrong guesses / minute before lockout

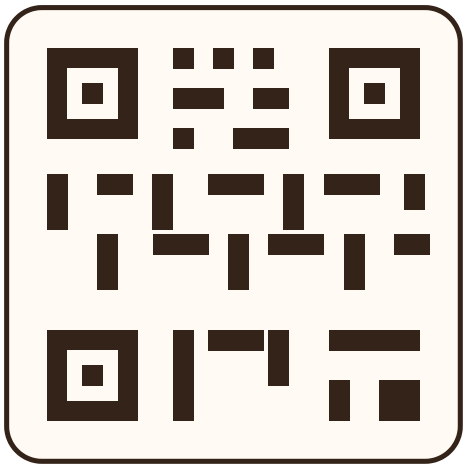

Figure 2 · The QR carries the camera's local IP, port, and the 48-bit pairing secret — nothing else.

## Why 48 bits is enough

A 48-bit secret has roughly **$2.8 \times 10^{14}$** possible values — about 281 trillion. Even against a hypothetical attacker on your local Wi-Fi running guesses as fast as the camera can answer (one every few milliseconds), they would still average tens of thousands of years of nonstop guessing before stumbling on the right value — and the rate limit below blows that out by many more orders of magnitude.

**BRUTE FORCE, IN PLAIN NUMBERS**

At the rate limit of 50 wrong guesses per minute, an attacker who never gives up would need about **5.4 MILLION YEARS**

on average to find the secret. The first humans appeared roughly 300,000 years ago, so the attacker would have had to start guessing about eighteen times before our species existed.

## Rotation

The secret automatically expires **90 days** after it is generated. If your QR code screenshot ever ends up somewhere it shouldn't — chat backups, screenshots, support emails — it is invalid after that window even before you notice. You can rotate it sooner by restarting Camera Mode at any time.

03 · TRANSPORT

# What encryption protects the stream

Once two phones agree on the pairing secret, they hand off the actual video and audio to WebRTC, the same peer-to-peer technology behind major browser-based video-call services.

## Signalling

The short handshake that exchanges connection parameters runs over a plain TCP socket on your LAN. It is authenticated by the 48-bit secret on every request, and the camera will refuse any unauthenticated message.

## Media

The actual stream is carried in **SRTP packets inside a DTLS session**. WebRTC makes this mandatory — there is no unencrypted mode. Both phones generate fresh DTLS keys on every session, so a recording of one session cannot be replayed against another.

**AES-GCM** cipher

**DTLS 1.2+** handshake

**ICE** connectivity

**Fresh keys / session** forward secrecy


Figure 3 · The media stream is encrypted on the camera phone and only decrypted on the monitor phone. Anyone in between sees ciphertext.

### WHY WE DON'T ADD A SECOND LAYER

DTLS-SRTP is already the security model that Apple, Google, Microsoft and Mozilla agree on for live video. Adding our own on top would only add risk, not safety.

04 · SERVER DEFENCE

# What stops a stranger on the same Wi-Fi

Coffee-shop Wi-Fi exists. Apartment building Wi-Fi exists. The pairing secret would still protect you on a hostile LAN, but we layer a few more defences in case someone tries to probe the camera anyway.

## Per-peer rate limiting

The camera tracks failed authentication attempts per source IP in a sliding 60-second window. After **50** wrong attempts in that window, the peer is blocked for **20 minutes** regardless of subsequent guesses.

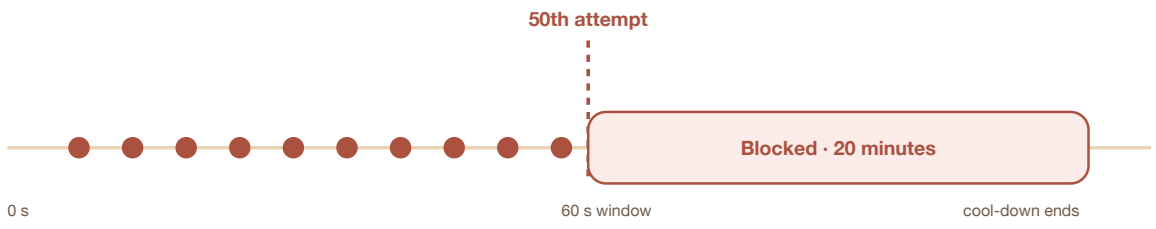


Figure 4 · One peer can spend up to 50 attempts inside any rolling minute. A bot doing one per second hits the cap in under a minute.

## Request size limits

- The TCP request buffer is capped at **1 MB**. A peer that opens a connection and dribbles bytes in slowly cannot keep the camera waiting forever.
- A POST body is capped at **512 KB**. A WebRTC offer is rarely more than 30 KB, so this cap is generous but still rules out the "send Content-Length: 999999999" attack.
- Malformed HTTP headers or non-numeric Content-Length values are rejected with a 400 response and the socket is closed.

## What an attacker on the same LAN actually has to do

Even with perfect timing and the entire 60-second window full of attempts every minute, an attacker needs to win the lottery on a 48-bit secret. The rate limit slows them down by a factor of millions; the entropy of the secret itself slows them down by a factor of trillions.

05 · PRIVACY

# What we never see, never store, never sell

TinyWatch is an unusual app in 2026: there is genuinely no account, no analytics tied to you, no server collecting your stream. The architecture above is also a privacy stance.

- **No account.** You never type an email or password. There is no user record for us to lose in a data breach.
- **No cloud video.** Your video and audio cross your Wi-Fi or your Personal Hotspot, never our infrastructure. We could not show you a clip from your camera if we tried.
- **No SSID exfiltration.** On iOS we ask for the Wi-Fi name only to show it on the reminder screen so you can confirm both phones are on the same network. It stays on the device.
- **No pairing-secret leakage.** The secret never leaves the two phones. It lives inside the QR code on the camera phone and in the monitor phone's memory while a session is active. Nothing else ever sees it.
- **Minimum-viable telemetry.** We use product analytics only for aggregate install / launch / crash counts, never for content or per-user behaviour inside a session. See the privacy policy for the exact list of fields.

## iOS permissions, justified one by one

**Camera** capture and stream live video

**Microphone** capture and stream live audio

**Local Network** find your paired phone on Wi-Fi

**Wi-Fi Information** display the network name on the reminder card

**Location (when in use)** required by iOS to read the Wi-Fi name

Permissions are requested when the feature that needs them is about to start, not on first launch. If you deny one, the app explains exactly which one and how to grant it again.

06 · HONESTY

# What TinyWatch deliberately does not protect against

A short, honest list. We would rather you know up front than discover later.

### SHARING YOUR QR WITH SOMEONE

Anyone you hand the QR screenshot to has the same access to the camera that you do, for as long as the 90-day secret is valid. Restart Camera Mode to invalidate it immediately.

### COMPROMISE OF YOUR WI-FI PASSWORD

TinyWatch assumes whoever is on your Wi-Fi already trusts the network. If your home router password is shared widely, every device with that password can at least *see* the camera exists, even if they cannot stream from it.

### COMPROMISE OF THE PHONE ITSELF

TinyWatch is an app, not a hardened device. Someone with physical access to an unlocked camera phone can do anything they want on that device, including viewing the live stream through the app's preview.

### LONG-RANGE OR AWAY-FROM-HOME USE

By design, TinyWatch only works when both phones are on the same local network. We do not cover the case of watching the camera from a different city — there is no relay server, no port forwarding setup, no remote-access mode. This is what makes the privacy story possible.

## If something goes wrong

TinyWatch has no telemetry that lets us see your camera or sessions remotely. If you want to report a problem, contact the addresses listed in the privacy policy and terms — those are the only channels through which we can help.