KEYSEE○: Deterministic Visual Identity for Public Keys

Scott Motte
scott@dotenvx.com
www.keysee.io

Abstract. Public keys are designed for machines, not people. They are long, high-entropy strings that provide strong cryptographic identity while offering almost no human affordance for recognition. Existing systems address this with fingerprints, word lists, QR codes, identicons, and random art, but these tools are usually either too textual, too utilitarian, or too visually weak to become a durable identity surface. We propose KEYSEE○, a deterministic visual identity system for compressed public keys. A public key is mapped into a 3D key form, visible serial mark, and renderable 2D image with an explicit 39-bit visual identity budget. The result is not cryptographic proof. It is a human-facing visual layer: stable, aesthetic, shareable, and useful anywhere public keys need to be recognized quickly.

1. Introduction

Modern cryptographic systems depend on public keys. A public key can identify a wallet, encrypt a message, verify a signature, or address an agent. Yet the public key itself is not a usable visual object. It is typically represented as a compressed hexadecimal string, such as *034ed7...*, whose correctness matters deeply but whose appearance is mostly noise to a human reader.

Software therefore has a visual identity gap. We routinely attach avatars, icons, names, badges, and logos to accounts and products, but cryptographic identity is still shown as text. The stronger the underlying identity becomes, the less human the representation tends to feel.

KEYSEE○ begins from a simple premise: public keys deserve an aesthetic surface. A visual identity system should be deterministic, so the same public key always produces the same image. It should be compact enough to render in UI and API contexts. It should be attractive enough that people actually want to use it. And it should be honest about what it is: a recognition layer, not a replacement for cryptographic verification.

2. Prior Visual Systems

There is a long history of making cryptographic or high-entropy identifiers easier for humans to compare. Each approach solves part of the problem but leaves important gaps.

2.1 Hex Fingerprints. Hex fingerprints are precise and universal. They are also visually hostile. A trained user can compare prefixes and suffixes, but most people cannot reliably distinguish long hexadecimal strings at a glance. Fingerprints are excellent for exact verification and poor as ambient identity. PGP fingerprints and OpenPGP key IDs are canonical examples of this tradeoff. [2][3]

```
SHA256:4b:8f:13:aa:91:5c:e7:40:2d:9f:8a:61:03:de:44:bc
03a34f8c2f6e8d90b31e14c5a6f09dd7a2f73c0b9e4d1886af91e2c44b75a6f1
```

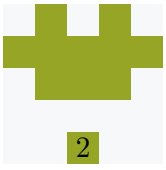
2.2 Word Lists and Safety Numbers. Word-based fingerprints and safety numbers improve verbal comparison. Signal-style safety numbers and QR verification are especially good for deliberate verification ceremonies. Their weakness is that they are episodic: they work when two people intentionally stop to compare identity, not when an interface needs a persistent, recognizable mark. [4]

```
atlas  ember  harbor  linen  quartz  violet
```

2.3 SSH Randomart. OpenSSH randomart turns a key fingerprint into ASCII art. Its Drunken Bishop algorithm was devised and implemented by Alexander von Gernler during OpenBSD Hackathon 2008. It is deterministic and compact, but its visual vocabulary is constrained by terminal characters. It is useful to specialists and memorable in a narrow context, yet it does not translate naturally into product surfaces, social previews, or brand-quality visuals. [1][9]

```
+--[ED25519 256]--+
|       .o+..     |
|       .o.+      |
|      ..o+ .     |
|     . +o.=      |
|    . o S+o.     |
|     + =.*o      |
|    . B.E+=      |
|     o.O+*       |
|     .+o+o.      |
+----[SHA256]-----+
```

2.4 Identicons, Blockies, and Jazzicons. Identicons and Ethereum-style blockies made deterministic visual identity common in developer tools and wallets. They are practical, fast, and recognizable as a category. But they often read as placeholder graphics rather than meaningful objects. Their grid-based aesthetics are useful but generic; they rarely carry the physical, material, or symbolic qualities that make a mark feel owned. [5][6][7]



KEYSEE○ takes the deterministic premise from these systems but changes the visual target. Instead of generating an abstract avatar, it generates a key: a familiar physical metaphor for access, identity, custody, and control.

3. Requirements

A public-key visual identity system should satisfy five requirements.

1) **Deterministic.** The same public key must always produce the same visual identity.
2) **Human-recognizable.** The output should contain shapes, proportions, and marks that can be compared quickly.
3) **Aesthetic.** Visual identity is not only information. It is trust, memory, and taste. A system that looks disposable will be treated as disposable.
4) **Explicit about entropy.** The visual identity space should be described mathematically so users understand what is being represented.
5) **API-native.** The output should be available as 3D geometry, SVG, PNG, and metadata so it can move across products without reimplementation.

4. Model

KEYSEE○ accepts a compressed public key encoded as 66 hexadecimal characters. For common elliptic curve public keys, the first byte is *02* or *03*, followed by 32 bytes of x-coordinate data. The visual model consumes portions of this public key to produce both geometry and visible marks.

Component	Visual bits	Description
Key family	3	Eight high-level key families define the primary silhouette.
Family shape	19	Within a family, additional bits alter geometry: blade form, shoulder, teeth, cuts, and proportions.
Visible prefix mark	17	The leading six hex characters are displayed as a visible mark. For compressed public keys, the first character is always 0, the second is 2 or 3, and the remaining four are hexadecimal.

This creates a total visual identity budget of 39 bits.

3 bits + 19 bits + 17 bits = 39 bits
2^39 = 549,755,813,888 possible visual identities

The visible prefix contributes 17 variable bits, not 24. The first character contributes

0 bits because it is always *0*. The second contributes 1 bit because it is *2* or *3*. The remaining four hex characters contribute 16 bits.

5. Rendering

The renderer produces a mesh, marks, and image outputs from the same deterministic payload. The 3D view uses a metallic key form so the identity has object-like presence rather than appearing as a flat avatar. The SVG and PNG views provide portable renderings for API consumers, Open Graph images, and UI surfaces.

The key includes two classes of marks. The front mark displays the public-key prefix in split serial form. The back mark contains the **KEYSEE**○ stamp. These marks are part of the identity surface but are not intended to replace the public key itself. They make the object legible while keeping the full key available in data and API responses.

```
GET /api/02f8fe5c795f926028bb4820665da651ca08ea2d640ceff86a9c94e93275cfe686.json

{
"geometry": {
"attributes": {
"position": {
"itemSize": 3,
"count": 3048,
"sample": [-0.1795, 1.3447, -0.0495, ...]
},
"normal": {
"itemSize": 3,
"sample": [0, 0, -1, ...]
}
},
"boundingBox": {
"min": [-0.6756, -1.7049, -0.0495],
"max": [0.6756, 1.7049, 0.0495]
}
},
"marks": {
"front": { "top": "02F", "bottom": "8FE" },
"back": { "text": "KEYSEE\u2394" }
}
}
```

6. Collision Analysis

KEYSEE○ does not attempt to visually encode every bit of the public key. It intentionally projects the key into a smaller visual identity space. With 39 visual bits, there are approximately 550 billion possible visual identities.

For random public keys, the approximate probability of at least one visual collision among n keys is:

$$p \approx 1 - e^{(-n(n-1)/(2N))}$$
$$N = 2^{39}$$

The 50% collision point is approximately:

$n \approx \sqrt{2N \ln 2} \approx 872{,}000$ keys

At 1 million keys, the chance of at least one visual collision is about 60%. This is acceptable for a recognition layer but not for proof. The visual identity is a fast human affordance. Exact cryptographic identity remains the public key.

7. Security Boundary

KEYSEE○ is not a security protocol. It does not authenticate a key, prevent substitution, or prove ownership. It is best understood as visual metadata derived from a public key.

This boundary is important. A visual key can help a person notice that something changed. It can make repeated exposure to the same key more memorable. It can support quick comparison in wallets, dashboards, agent registries, and documentation. But when exact trust is required, software should compare the full public key or a cryptographic fingerprint.

8. Aesthetics

Aesthetics are not decoration added after the technical work. They determine whether a system is remembered, trusted, and reused. A public key shown as raw text remains inert. A public key rendered as an object can become a mark.

The key metaphor is deliberately old. Keys are familiar symbols of access and custody. By using a metallic form, engraved marks, and a compact badge identity, **KEYSEE**○ gives cryptographic identity a visual language that is both technical and tactile.

The goal is not to make cryptography cute. The goal is to make it visible.

9. Conclusion

Public keys are foundational but visually poor. Existing identity aids improve comparison, but many remain textual, temporary, or aesthetically weak. **KEYSEE**○ proposes a deterministic visual identity system that maps compressed public keys into stable, shareable, object-like marks with an explicit 39-bit visual identity budget.

References

[1] OpenBSD, "OpenSSH VisualHostKey Randomart", https://man.openbsd.org/ssh_config#VisualHostKey.

[2] P. Zimmermann, "The Official PGP User's Guide", https://web.mit.edu/network/pgp.html, 1994.

[3] OpenPGP Working Group, "OpenPGP Message Format", https://www.rfc-editor.org/rfc/rfc4880, 2007.

[4] Signal, "Safety Numbers", https://support.signal.org/hc/en-us/articles/360007060632-Safety-Numbers.

[5] GitHub, "Identicons", https://github.blog/news-insights/product-news/identicons/, 2013.

[6] Ethereum Foundation, "Ethereum Blockies", https://github.com/ethereum/blockies.

[7] MetaMask, "Jazzicon", https://github.com/MetaMask/jazzicon.

[8] Unicode Consortium, "The Unicode Standard", https://www.unicode.org/standard/standard.html.

[9] D. Loss, T. Limmer, and A. von Gernler, "The drunken bishop: An analysis of the OpenSSH fingerprint visualization algorithm", https://www.dirk-loss.de/sshvis/drunken_bishop.pdf, 2009.