Whisper Security

technical report · *printed-channel brand protection*

Preprint · reproducible · CC BY 4.0



Cinnamomum verum, true cinnamon. Köhler's Medizinal-Pflanzen, plate 182 (1887). Public domain; cyanotype-toned reproduction. In commerce it is routinely passed off by Cinnamomum cassia, a confusable the buyer cannot tell apart without chemistry.

PREPRINT

Tattooing the Root Zone

QR-Inking: moving a printed QR to a different real top-level domain

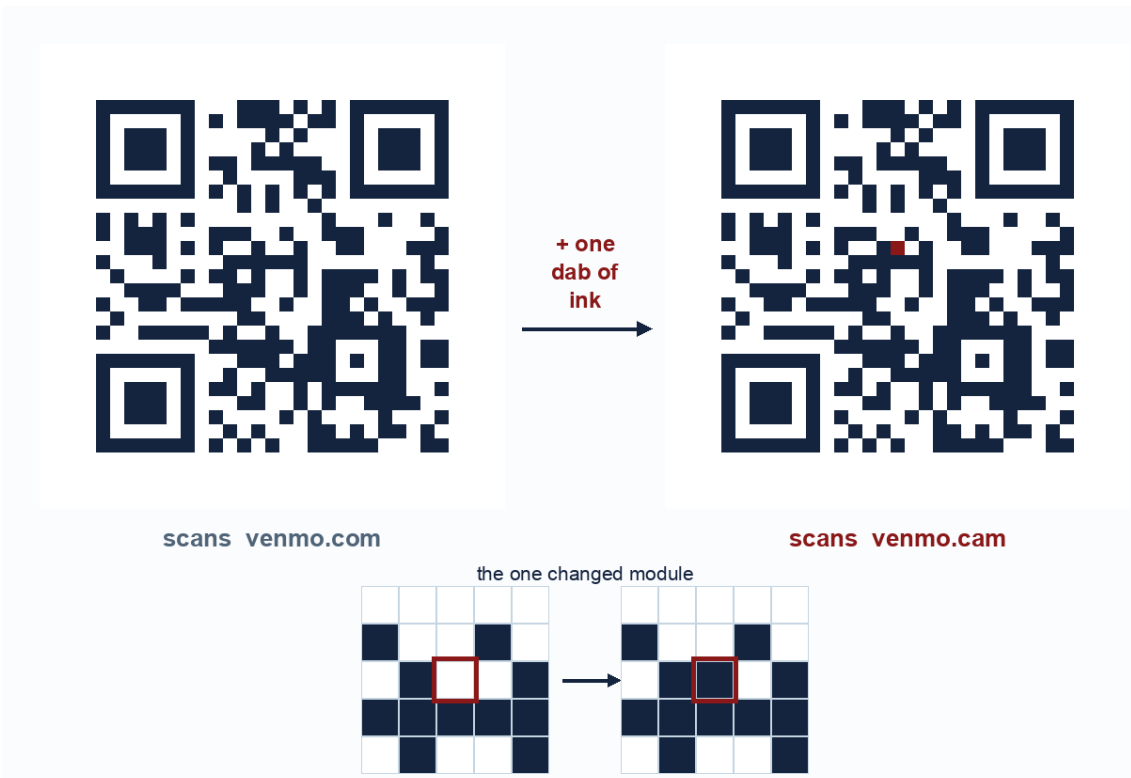
Kaveh Ranjbar[1]

Whisper Security

November 16, 2026

Every number in this paper is regenerated and checked by the committed harness `src/validate.py` *(166 of 166 checks pass in this build).*

[1]Kaveh Ranjbar is the co-founder and CEO of Whisper Security. He was at the RIPE NCC for fifteen years, including operating the K-root name server; served six years on the ICANN Board; chaired the IETF Trust; and holds an MSc from the University of Oxford. Correspondence: `kaveh@whisper.security`.



A printed QR code is ink. Add one dab, and it resolves to a different real top-level domain.

PREPRINT

Abstract

A printed QR code is a Reed-Solomon codeword rendered in ink, and ink can be edited. We show that a QR symbol can be printed so that it decodes to a legitimate host A and sits one *directed* module-flip from a same-length host B: one dab of ink added to a pre-printed *boundary* symbol changes the decoded host, while the un-inked symbol still scans as A. The method that finds and ranks these edits we call *QR-Inking*. The dangerous neighbours are not typos but *other real top-level domains*. We prove that, for a given same-length TLD pair, the *module distance* of the morph depends only on the length of the second-level label and is mask-independent, so the exposure is a property of the IANA root zone. Enumerating all 1436 delegated TLDs, we find it dense: among the 248 two-character TLDs alone there are 1881 one-module morph pairs, many a single added dot between TLDs an attacker can register (`.ai` to `.am`, `.net` to `.new`). For ASCII morphs the coding and visual channels are orthogonal; the one regime where they combine is internationalised TLDs, where a simplified-form Chinese ccTLD becomes its traditional variant in two modules and, at URL-bar size, the preview reads as the same word. We validate on two independent decoders and a phone camera, give a detection test that follows from the construction, and regenerate every number with a committed harness.

> **QR-Inking and TLD morphing, the headline result**
> A printed QR symbol can decode to host A and sit one *directed* module-flip from a same-length host B, while the un-inked symbol still scans as A and the same flip on a stock symbol retargets nothing. When A and B differ only in the TLD, the *module distance* depends, for a given TLD pair, only on the length of the second-level label (not its text, and not the mask), so the exposure is a property of

the root zone. Among the 248 two-character TLDs there are 1881 one-module and 5647 two-module morph pairs (any polarity). The dangerous subset is the registrable open-to-open morphs, where the attacker can take the name over: a single *added dot* turns `.ai` into `.am` and `.net` into `.new`. For internationalised TLDs the two channels combine: a simplified-form Chinese ccTLD becomes its traditional-character variant in two modules, decode-verified on two decoders, and the preview shows the same word. The boundary print carries $t = \lfloor n_{ec}/2 \rfloor$ preloaded symbol-errors toward B; this is not a claim that an ordinary QR code is one dot from a lookalike.

1. Introduction

A QR code is a printed instruction that a human cannot read. It appears on posters, parcels, restaurant tables, parking meters, utility bills, and payment stands, and modern phone cameras do surface the decoded host in a preview before navigating, so the viewer *can* read it. But most do not: in a 2024 naturalistic study, 67% of people who scanned a code opened the link without inspecting the URL [15].

That makes a printed QR a trust channel in the sense a domain name is, with one extra weakness: the bytes are rendered as ink, and ink can be edited after printing.

This paper asks how small an edit suffices to change the host a QR code resolves to, and answers it precisely.

The stakes are concrete. In China a printed QR carries the large majority of mobile payments, on the order of 85 to 90 percent, across hundreds of millions of merchants, many of them street vendors with a single static code taped to a counter [19]. Southeast Asia runs ten national QR schemes with eight cross-border links and a multilateral blueprint on top [21]. On that surface physical tampering already steals money: swapping the sticker on a merchant's payment code is a documented, prosecuted fraud [20], and the FBI has warned that criminals tamper with QR codes to redirect funds [18].



Cinchona pubescens, the Andean bark that yields quinine. Köhler plate 161. A counterfeited medicine for centuries: the bark looks right, and only an assay tells the drug from the filler. A QR code is the same kind of object, a mark the eye cannot audit.

A plain sticker swap is the crude version, and the field's answer to it is a controlled print on a tamper-evident or no-sticker surface, re-inspected at an acceptance scan (Section 2). QR-Inking is the edit that survives those answers: it is the printed code itself moved by a few ink modules, so there is nothing to peel, and the morph passes the inattentive scan that most people give a code [15]. A reader who does inspect the preview of an ASCII morph sees a *different* name; the one regime that defeats that reader too is the internationalised look-alike (Section 5.5), where the Chinese simplified and traditional forms read as the same word. Section 5.6 puts real payment and account hosts one or two inked modules from a registered or registrable look-alike.

The naive fear, that a QR code is one dot away from a lookalike, is false. A QR symbol is a Reed-Solomon codeword. Two different same-length URLs, encoded at the same version and error-correction level, are far apart as codewords: in every block their difference touches, they differ by at least $d_{min} = n_{ec} + 1$ symbols. On the running example, `https://google.com` and

PREPRINT

`https://googlg.com` differ by 17 symbols and 54 modules at version 2, level M. One module on a normally generated symbol cannot cross that distance.

> *The fear that any QR code is one dot from a lookalike is false. The useful truth is narrower, and worse: a code the attacker prints can be one directed dot from another real top-level domain.*

What is true is sharper and more useful to a defender. An attacker who controls the *print* can lay down a third symbol: a *boundary* print that still decodes to the legitimate host A, because it sits exactly on the correction radius $t = \lfloor n_{\mathrm{ec}}/2 \rfloor$, with its t preloaded symbol-errors aimed at one chosen neighbour B. A single directed edit of the remaining modules then crosses the radius, and the code decodes to B. The edit can be constrained to one physical operation: adding ink with a pen or a sticker, or removing it with a scratch. We name the method that finds and ranks these edits *QR-Inking*.



Coffea arabica. Köhler plate 189. Blended at the wholesale level with the cheaper Coffea canephora; the cup is detectable only by someone looking for it. Our detection test (Section 7) is the same idea: the boundary print is already sitting on the correction radius, and a decoder that reports its corrections sees it.

The central contribution is where this construction bites hardest. The same-length neighbours that matter are not misspellings of the host; they are *other real top-level domains*. We prove that the any-polarity module cost of a same-length TLD morph is content-independent: it depends only on the length of the second-level label, not its text. We then measure the entire IANA root, all 1436 delegated TLDs as of Version 2026111500, and map which TLDs are a cheap directed edit from which. The exposure is dense, it is structural, and in many cases both endpoints are TLDs an attacker can register today.

Contributions.

- *A printed trust channel.* We formalise the QR code as an editable printed codeword and separate the false claim (an ordinary QR is one dot from a lookalike) from the true one (a pre-printed boundary symbol is one directed dot from a neighbour).
- *The QR-Inking construction.* A per-block boundary build over the Reed-Solomon structure of ISO/IEC 18004, with a directed (add-ink or remove-ink) edit, and a ranking that returns the cheapest same-length host conflicts in a chosen direction (Section 4).
- *An enumeration and analysis of the IANA root.* Content-independence of the morph cost, and an exposure map over all 1436 TLDs grouped by length, including a table of high-value, easy-to-ink, registrable morphs and the restricted namespaces the construction reaches (Section 5).
- *Detection and validation.* A detector that follows from the construction, a decode validation on two independent decoders and a phone camera, and a committed harness that regenerates every number in this paper (Sections 6 and 7).

PREPRINT

2. Threat model

Where this is deployed. The setting is the printed static code: a payment counter card, a restaurant table tent, a vending-machine panel, a parking meter, a utility bill. These are the acceptance technology for the national QR payment schemes, China's Alipay and WeChat Pay and the Southeast Asian schemes (Indonesia QRIS, Thailand PromptPay, Singapore PayNow and SGQR, Malaysia DuitNow, Cambodia KHQR, Vietnam VietQR) and their cross-border links [21]. A static counter code is printed once and then trusted for months, which is the window a boundary print or a field ink edit exploits.

Attacker. The attacker controls the printed symbol before it reaches the victim and can make one small, physical ink edit in the field. Two capabilities are distinguished. First, printing a boundary symbol: a poster, sticker, parcel label, or payment-stand card that the attacker produces, which decodes to the legitimate host A (we verified this on two software decoders and, for two examples, a phone camera; Section 6). Second, a field edit: adding a small amount of ink with a pen or an adhesive dot, or removing ink by scratching. Adding ink is the realistic field move, since it needs no precision tooling and leaves the rest of the symbol intact.

Goal. Retarget the decoded host from A to a chosen same-length host B with the smallest possible edit, so that the symbol reads A before the edit (for example at an acceptance scan) and B after, with the edit a speck or a scratch rather than an obvious overprint. The boundary print has zero error-correction margin (Section 6), so this robustness is at the level we actually demonstrate, two software decoders and two field flips, not a surviving-everything claim.

Why not just a sticker? The dominant real-world technique is stickering a fully-malicious QR over a legitimate one, which needs no coding theory and decodes reliably. QR-Inking's only advantage is the boundary property: the un-edited symbol reads A and passes a scan, and a single later ink edit retargets it to B. That matters only in a workflow where the code is scanned and approved as A, then edited in the field, and not re-inspected: an acceptance scan or proofing step that checks the code reads A at production, a tamper-evidence seal that a sticker would break but a pen dot would not, or a surface that will not take a sticker but will take ink. Where none of these holds, a sticker is the better attack, and we say so.

In scope. Same-length retargeting through a pre-printed boundary symbol plus a directed module edit; both ASCII hostnames (Section 4) and, as our core case, same-length TLD swaps between real delegated TLDs (Section 5).

Out of scope. Reprinting the whole symbol (trivial, but it is a full substitution, not the minimal boundary edit we study, and it forfeits the property that the un-edited symbol reads A); length-changing edits such as a TLD of a different length, an inserted dot, or an added label, which move the byte-length header and are a different distance (Section 4); and compromise of the generator or the scanner software. We also do not claim the morph targets are visually confusable or already registered; where it matters, we state that both endpoints are real delegated TLDs, which is the point.

A clean code is not one dot away. An ordinary QR code, printed by a normal generator, is *not* one dot from a lookalike host. The attack requires the attacker to print the boundary symbol. The contribution is to show how cheap the final edit then is, how it can be constrained to a single physical operation, and how much of the DNS root zone it reaches.

> *A clean QR code is safe. The boundary symbol is the attack: a pre-printed code that reads as the real host until one dot of ink is added.*

PREPRINT

3. Background

Byte mode and the header. Public URL QR codes use byte mode. On versions 1 through 9 the header is a 4-bit mode indicator (`0100`) and an 8-bit character count, twelve bits, followed by the raw ISO-8859-1 bytes, a terminator, padding bits to a codeword boundary, and the alternating pad codewords `11101100` and `00010001` to the data capacity. Twelve is not a multiple of eight, so every payload byte straddles two data codewords: character k occupies bits $[12 + 8k,\ 12 + 8k + 8)$. A one-character edit therefore touches two data symbols, each carrying its Reed-Solomon parity with it.

Reed-Solomon and MDS distance. Within each block the data codewords are extended by n_{ec} parity symbols over GF(2^8), the field with primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$ (`0x11D`), using the generator $g(x) = \prod_{i=0}^{n_{\text{ec}}-1}(x - \alpha^i)$ with $\alpha =$ `0x02` and first root α^0. The code is maximum-distance-separable: any two distinct codewords of a block differ in at least $d_{\min} = n_{\text{ec}} + 1$ symbols. A bounded-distance decoder of radius $t = \lfloor n_{\text{ec}}/2 \rfloor$ returns the unique codeword within t symbols of the received word, and since $2t < d_{\min}$ no received word lies within t of two codewords. For version 2 level M, $n_{\text{ec}} = 16$, so $d_{\min} = 17$ and $t = 8$.

Blocks, interleaving, masks, format. From the larger versions and levels the message is split into several Reed-Solomon blocks, interleaved per ISO/IEC 18004 clause 7.6. A mask XORs a fixed pattern over the encoding region to balance the image; it changes which modules are dark and light but not how many codeword bits differ between two messages. Format information records the level and mask, so a decoder knows t. We verify our encoder against segno 1.6.6 at the module level (Section 6).

4. The QR-Inking construction

4.1 *Boundary symbols and directed edits*

The construction runs independently in each Reed-Solomon block, which is what makes it apply past the single-block codes. Let D be the set of symbols in which the codewords for A and B differ within a block. Choose a user set $U \subseteq D$ and a preload set $P \subseteq D \setminus U$ with $|P| = t$ and $|U| = |D| - 2t$. Print A's codeword with the symbols in P replaced by B's values; leave the rest of D equal to A. The printed block is then at symbol distance t from A, so the decoder returns A; it is at distance $|D| - t$ from B, which is $t + 1$ when $c_A \oplus c_B$ restricted to the block is a minimum-weight codeword ($|D| = d_{\min}$) and n_{ec} is even. The editor then inverts exactly the modules of U: the block moves to distance $t + |U|$ from A and distance t from B, and the decoder returns B. When $|U|$ is one symbol with a single bit set, the whole edit is one module.

PREPRINT

```
RankConflicts(url, version, level, masks, polarity, n):
  best = {}
  for mask in masks:
    A = encode(url, version, level, mask)        # byte mode, RS blocks, mask applied
    for target in candidates(url):               # same-length host or TLD edits
      plan = Boundary(A, target, polarity)        # skip if this direction is impossible
      if target not in best or key(plan) < key(best[target]):
        best[target] = plan
  return the n plans with the smallest keys

Boundary(A, B, polarity):                        # per Reed-Solomon block
  for each block whose difference set D is non-empty:
    t = floor(n_ec / 2);  u = |D| - 2t
    reject if u < 1                              # distance is inside the radius
    allowed = { s in D | every module of s has `polarity` on the print of A }
    reject if |allowed| < u
    U = the u lightest symbols in allowed;  P = any other t symbols of D
  print A with each symbol in P replaced by the same symbol of B
  the user inverts only the modules belonging to U
```

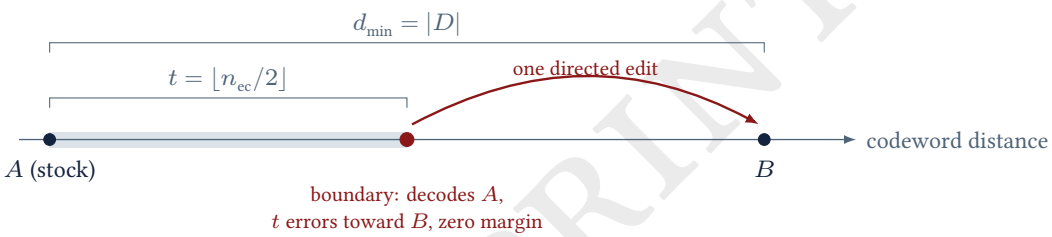


Figure 1. *The construction as a codeword distance line. The shaded ball is A's correction radius t. The printed boundary symbol sits at distance exactly t from A (it decodes A, carrying its t preloaded errors toward B, with no margin to spare); B is at distance $d_{\min} = |D|$. One directed edit of the user symbols crosses the radius, and the decode flips to B.*

Direction, which we call polarity, is a property of the printed module, not of the codeword bit. The symbols of U are not preloaded, so their colour on the boundary print is their colour on the stock print of A. A light module (0) can only be turned dark (add ink); a dark module (1) can only be turned light (remove ink). A symbol whose difference spans two bits is usable one-way only when both modules face the same way. The mask does not change how many bits differ; it changes their colour, because the printed module is the codeword bit XOR the mask. Choosing the mask is how an unconstrained one-module conflict is assigned to add-ink or remove-ink.

4.2 *Ranking*

QR-Inking is the ranking, not a single pair. Given a legitimate URL, a version, a level, the masks to try, a polarity, and n, it returns the n cheapest same-length conflicts, fewest user modules first. The candidate generator is pluggable; a confusable filter, a registration check, or the same-length TLD set of Section 5 can be substituted without changing the ranking rule.

PREPRINT

4.3 *Worked example one: a host typo (google.com)*

Figure 2 is the claim for the cheapest unconstrained edit of `https://google.com` at version 2, level M, mask 0. The boundary symbol (centre) still decodes `google.com`; clearing one dark module at $(15, 15)$, a bit of the last letter of `google`, decodes `googlg.com` (right); the same dot on a stock symbol (left) changes nothing.

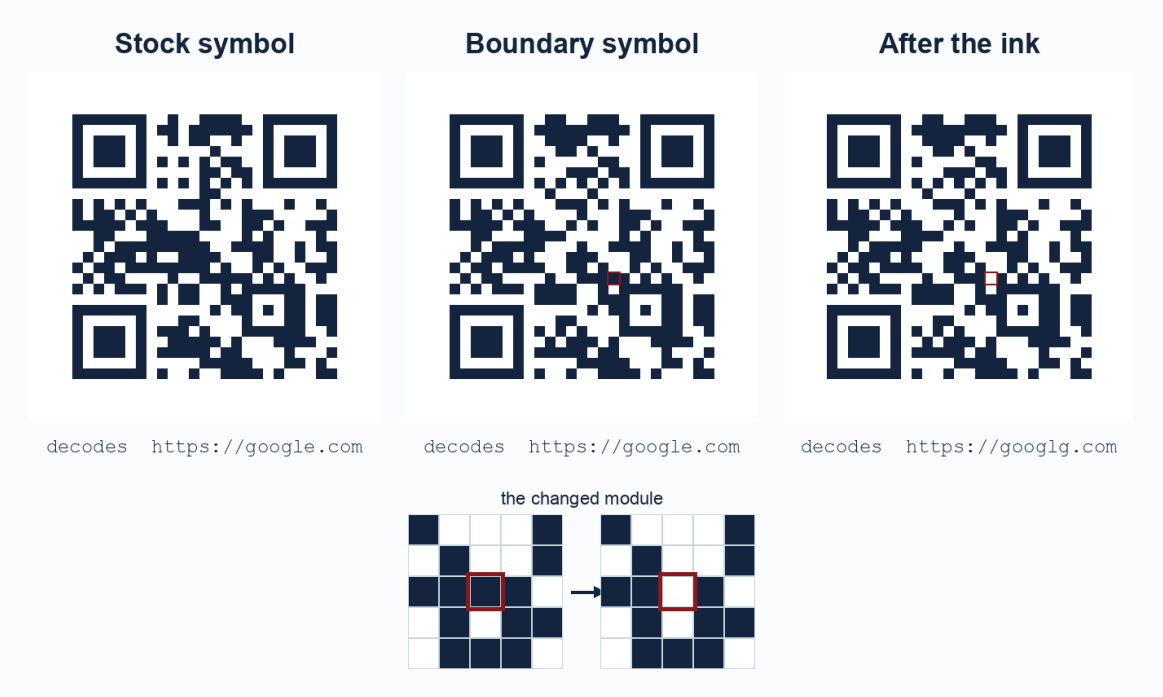


Figure 2. *Version 2, level M, mask 0. Left: stock symbol for* `https://google.com`*. Centre: boundary symbol, still decoded as* `https://google.com`*, module* $(15, 15)$ *ringed. Right: that module cleared, decoding* `https://googlg.com`*.*

The lower bound is measured, not assumed. Converting whole symbols off a *stock* `google.com` symbol, lightest first, still decodes `google.com` after 8 symbols (18 modules); only the ninth symbol, which is $|D| - t$, returns `googlg.com` (Table 2). Direction is a mask choice: the same bit of `googlg.com` is a dark module under mask 0 (remove ink) and a light module under mask 1 (add ink), at the identical coordinate. Figure 3 shows this, with the cheapest add-ink conflict `qoogle.com` for contrast.

PREP

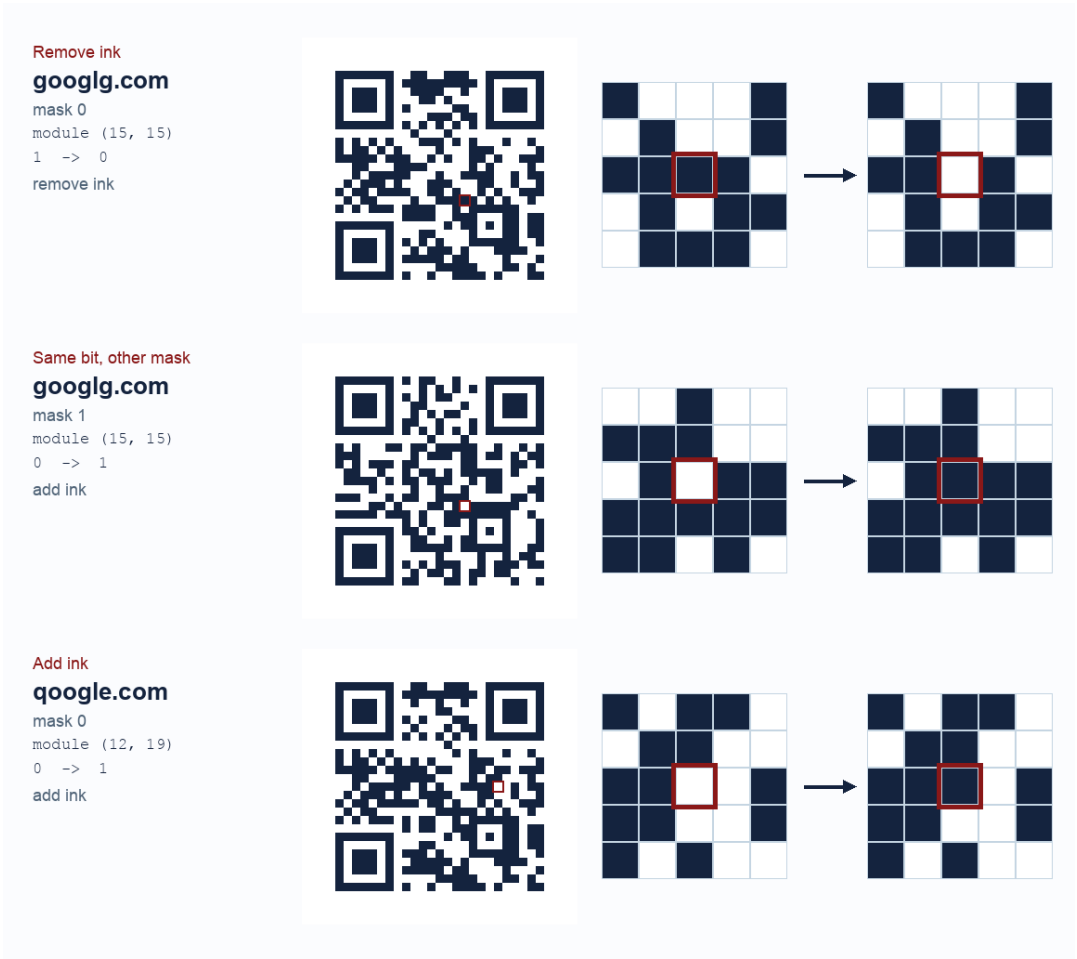


Figure 3. *Directed edits on a version-2 level-M boundary symbol, with a five-by-five inset around the user module. Top: remove ink at* $(15, 15)$*, mask 0, to* `googlg.com`*. Middle: the same coordinate under mask 1, now add ink. Bottom: add ink to* `qoogle.com`*, mask 0. Each left-hand code is the boundary print and still decodes* `google.com`*.*

Of the 361 one-character host edits at this version, 152 are at the minimum symbol distance 17 and 209 at 18; 51 reduce to a single hostname module once the symbol sits on the boundary. Table 1 is the ranking at mask 0, in all three polarities.

#	Either direction	Add ink $0 \to 1$	Remove ink $1 \to 0$
1	`googlg.com` (15,15)	`qoogle.com` (12,19)	`googlg.com` (15,15)
2	`googla.com` (15,16)	`googld.com` (14,16)	`googla.com` (15,16)
3	`google.cmm` (2,15)	`google.kom` (8,15)	`google.cmm` (2,15)
4	`google.cnm` (1,16)	`googlencom` (13,16)	`google.cnm` (1,16)
5	`gyogle.com` (9,17)	`googlm.com` (16,15)	`gyogle.com` (9,17)
6	`qoogle.com` (12,19)	`google.ckm` (2,16)	`6oogle.com` (10,19)
7	`6oogle.com` (10,19)	`google.co-` (0,16)	`googlu.com` (17,15)
8	`googld.com` (14,16)	`google.cvm` (3,16)	`goo6le.com` (24,17)

Table 1. *Top 8 conflicts for* `https://google.com` *at version 2, level M, mask 0. Each is one module on a hostname character; coordinates are (row, column) before the quiet zone.*

Symbols converted	Modules	Decodes as
1	1	`google.com`
8	18	`google.com`
9	21	`googlg.com`

Table 2. *Converting the lightest whole symbols off a stock* `google.com` *symbol. The decode switches only at* $|D| - t = 9$ *symbols, the crossing point implied by the minimum distance (*$d_{\min} - t$*, not* $d_{\min}$ *itself, which is 17).*

Figure 4 gives the cheapest edit for `google.com` across the versions and levels it fits. Every even-n_{ec} configuration is one module; version 3 level L has odd $n_{ec} = 15$, so $d_{\min} = 2t + 2$ and the cheapest edit is two modules. The same holds for the other hosts we tested: at version 2 level M, the one-module hostname-edit counts are 52 of 325 for `apple.com`, 62 of 397 for `example.com`, 52 of 361 for `paypal.com`, and 51 of 361 for `google.com`.

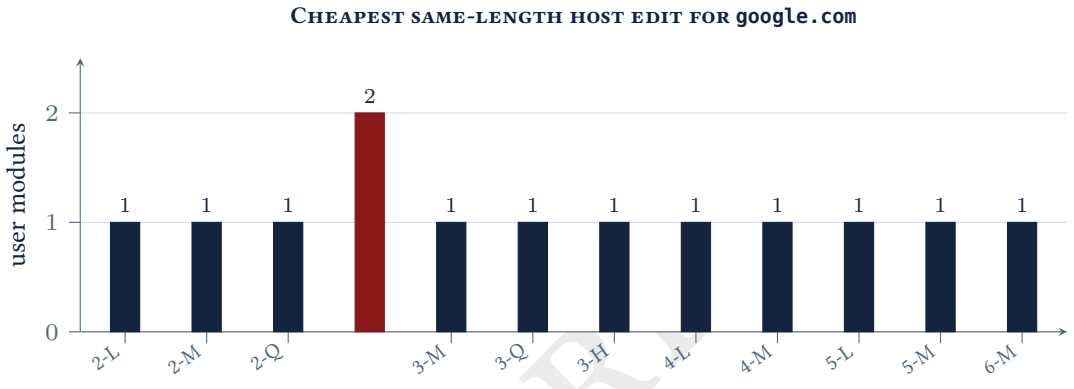


Figure 4. *Cheapest same-length host edit for* `google.com` *by version and level, mask 0. Even* n_{ec} *gives one module; odd* n_{ec} *(version 3 level L) gives two.*

4.4 *Worked example two: a TLD morph (kaveh.org)*

The second example is the author's own domain, and it introduces the paper's core move: morphing the TLD. segno's default encoding of `https://kaveh.org` is version 1, level L, where $n_{ec} = 7$ is odd, so $d_{\min} = 2t + 2$ and the cheapest same-length edit is 2 modules: the natural encoding of this domain is, by the arithmetic of its length, resistant to the one-module form. Printed instead at version 2, level M, `https://kaveh.org` morphs to `https://kaveh.ong` in 2 modules, and `.ong` is the real delegated TLD for non-governmental organisations (Figure 5). The boundary print and the stock symbol both decode `kaveh.org`; the two edited modules decode `kaveh.ong`. Table 3 lists the cheapest real-TLD morphs of `kaveh.org`; every target is a delegated TLD, and the one-module cases reach `.cbn` and `.msd`.

PREPRINT

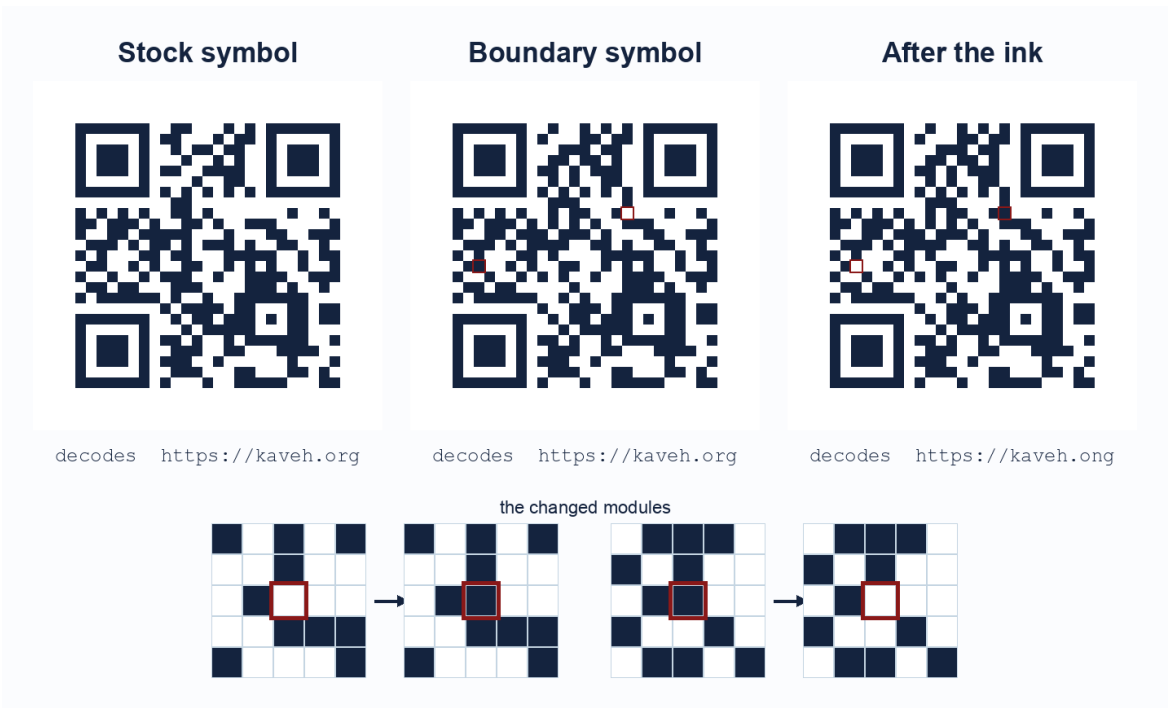


Figure 5. *`https://kaveh.org` at version 2, level M, mask 0. Left: stock symbol. Centre: boundary symbol, still `https://kaveh.org`, the two user modules ringed. Right: those modules edited, decoding `https://kaveh.ong`. `.ong` is the real NGO TLD.*

#	Modules	Target (real delegated TLD)
1	1	`kaveh.cbn`
2	1	`kaveh.msd`
3	2	`kaveh.abc`
4	2	`kaveh.aco`
5	2	`kaveh.aig`
6	2	`kaveh.bbc`
7	2	`kaveh.bmw`
8	2	`kaveh.jcb`
9	2	`kaveh.nba`
10	2	`kaveh.nrw`
11	2	`kaveh.ong`
12	3	`kaveh.aws`

Table 3. *Cheapest same-length TLD morphs of `https://kaveh.org` at version 2, level M, mask 0. Every target is a real delegated TLD.*

PREPRINT

5. TLD morphing across the IANA root

The host-typo case is a curiosity; the TLD case is a full enumeration of the root zone, because the second-level label can be held fixed while the TLD ranges over all 1436 delegations. It is a deterministic analysis of coding distances among fixed strings, not a measurement of in-the-wild behaviour, and we call it so. This section is the core of the paper.

5.1 *Two metrics, and a content-independence proposition*

We keep two quantities separate throughout. The *module distance* of a morph is the any-polarity, any-module edit cost: the minimum number of modules that differ between the printed boundary symbol and its edited form, with the editor free to add or remove ink anywhere. The *added-ink cost* is the directed quantity: the number of modules an attacker who may only add ink must darken, for a specific printed label and mask. The module distance is the structural quantity we enumerate over the root; the added-ink cost is what a field attacker pays, and it is instance-specific. The IANA counts are module distances; the "add N dots" examples are added-ink costs for a named label and mask. We never conflate them.

The module distance has a clean invariance, which is what turns a per-brand curiosity into a property of the root zone.

> **Proposition 1 (content-independence)**
> Fix byte mode, a QR version and level, and a same-length TLD pair (X, Y). For any two second-level labels L_1, L_2 of equal length, the morph $L.X \to L.Y$ has the same module distance whether $L = L_1$ or $L = L_2$, and that distance is the same under every mask.

Proof. In byte mode the header (the mode indicator and the 8-bit character count) and the pad codewords are determined by the total length, which is equal for $L_1.X$ and $L_2.X$ and for their Y counterparts. So for a fixed label the two data vectors differ only in the TLD byte positions, with the label-independent value $X \oplus Y$ there. The systematic Reed-Solomon parity map is $\mathrm{GF}(2^8)$-linear, hence $\mathrm{GF}(2)$-linear, so the parity difference is the parity of the data difference, $\Delta_{\text{par}} = \text{parity}(\Delta_{\text{data}})$, and is also label-independent. The whole codeword bit-pattern $\Delta = \Delta_{\text{data}} \,\|\, \Delta_{\text{par}}$, its per-symbol weight multiset, the difference-set size $|D|$, the user-set size $|U| = |D| - 2t$, and the sum of the $|U|$ lightest symbol weights are therefore all independent of the label. The module count equals this codeword bit-distance through the label-independent placement bijection; the mask is XORed into both the boundary print and its edited form, so it cancels in Δ and the count is mask-independent. ■

We check all three steps directly: for `com` to `cam`, `com` to `net`, `org` to `ong`, and `net` to `pet`, the module distance is 2, 6, 2, and 2 respectively, identical across three distinct six-character labels. The result holds *for a given TLD pair* at a fixed version and level; the cost still depends on which pair and which version. Directed added-ink costs are not content-independent, because whether a module is currently light depends on the printed label and the mask, so we report those for a named label and mask.

PREPRINT

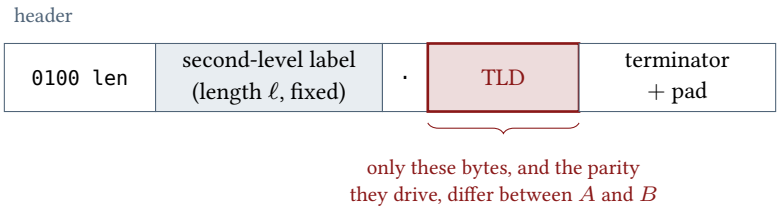


Figure 6. *The byte layout behind Proposition 1. The header and the pad are fixed by the total length; holding the second-level label length ℓ fixes the TLD byte positions, so between A and B only the TLD bytes (and, by linearity, the parity they drive) change. The difference pattern, and therefore the module distance, is the same for any ℓ-length label.*

5.2 *The IANA measurement*

We take the authoritative IANA root zone TLD list (Version 2026111500), 1436 delegated TLDs, 1285 ASCII and 151 internationalised. Grouping the ASCII TLDs by length and holding the second-level label at a representative length of six (version 2, level M, mask 0), we compute the module distance for every same-length pair; by Proposition 1 the counts are the same for any other six-character label and any mask, though they shift at other label lengths and versions. Figure 7 and Table 4 give the exposure. It is dense and it decays sharply with TLD length. The reason is a clean property of the code: a difference confined to a single information (message) symbol forces an MDS codeword to weight exactly $d_{\min}$ (if any parity symbol vanished the weight would be below $d_{\min}$), so at even n_{ec} such a morph has $|U| = |D| - 2t = 1$ automatically (at odd n_{ec} it is $|U| = 2$, as the Limits note), and whether it is one module then turns only on the leftover symbol's bit weight. A one-module morph needs $|D| = d_{\min}$ with the lightest difference symbol of bit weight one. The dominant way to meet this is a single-symbol data difference (a two-symbol difference can also reach $d_{\min}$ when one parity symbol vanishes), and it is what explains the length decay: shorter TLDs change fewer bytes, so a larger fraction of their morph pairs are single-symbol, which is why length two carries 1881 one-module pairs and length four only 39.

> *Shorter TLDs collide more: 1881 one-module morph pairs among the two-character TLDs, only 39 among the four-character TLDs.*

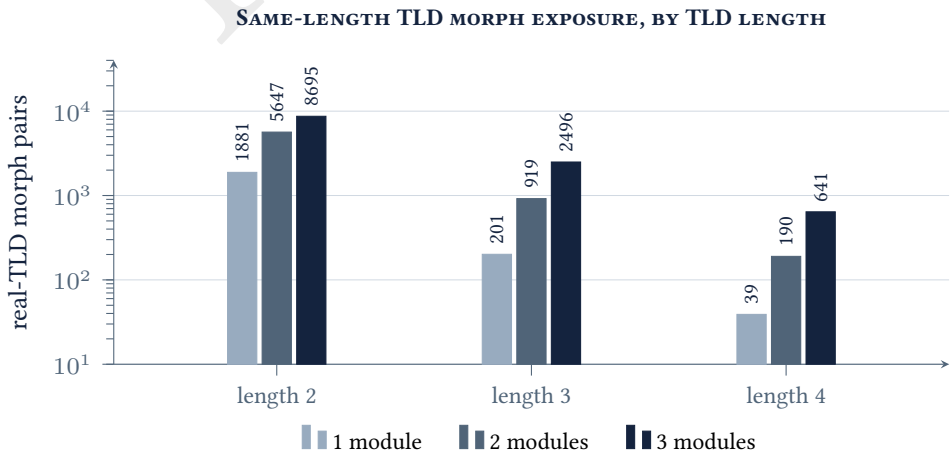


Figure 7. *Same-length TLD morph exposure across the IANA root: count of real-TLD pairs at one, two, and three modules, by TLD length (any polarity, label length six, version 2 level M). Log scale.*

PREPRINT

TLD len	TLDs	1 mod	2 mod	3 mod	Densest hub
2	248	1881	5647	8695	`.cg` (167 within 3)
3	216	201	919	2496	`.bet` (63 within 3)
4	205	39	190	641	`.care` (23 within 3)

Table 4. *Same-length TLD morph pairs by length. The densest hub is the TLD with the most other real TLDs within three modules.*

Among the 248 two-character TLDs there are 1881 one-module and 5647 two-module morph pairs; among the 216 three-character TLDs, 201 and 919; among the 205 four-character TLDs, 39 and 190. The exposure concentrates in hubs. Figure 8 ranks the most-exposed TLDs by how many other real TLDs sit within three modules. The densest two-character hub, `.cg`, reaches 167 of the 248 ccTLDs, and the `.c`-heavy block (`.cg`, `.cf`, `.cu`, `.cw`, `.cc`, `.cm`) all sit within three modules of roughly 160 of them. At length three the densest hubs are `.bet` (63 within three), `.cat`, and `.fit`; at length four, `.care` (23 within three) anchors a cluster with `.case`, `.cash`, `.casa`, and `.cafe`. Figure 9 draws one such cluster as a graph; a single TLD is a one-dot edit from several of its neighbours.

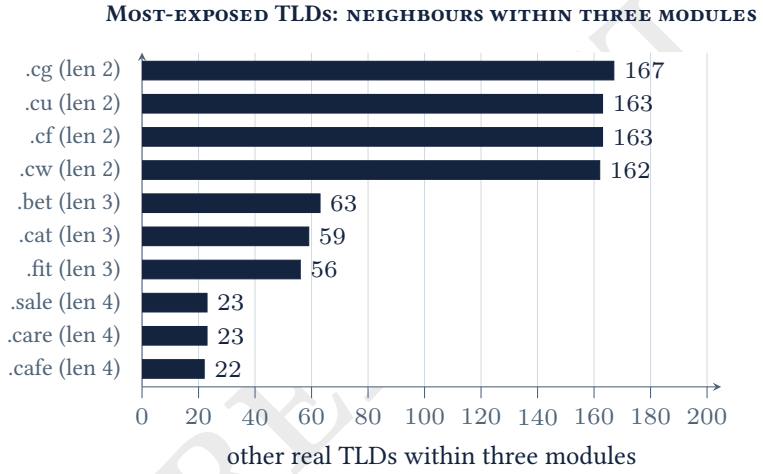


Figure 8. *Most-exposed TLDs per length class, ranked by the number of other real TLDs within three modules (any polarity, label length six, version 2 level M).*

PREPRINT

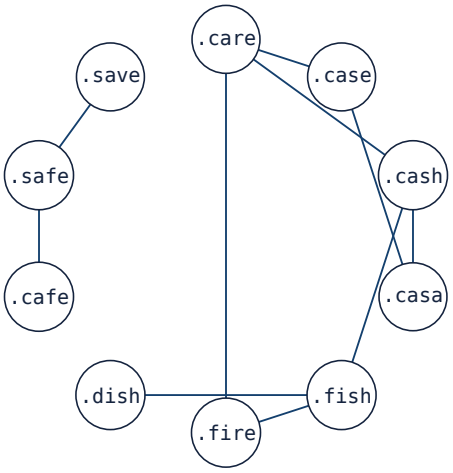


Figure 9. *A one-module cluster of real length-4 TLDs. Each edge is a single added or removed module between two delegated TLDs; a single TLD is a one-dot edit from several of its neighbours.*

Why the root is exposed, and why it is getting worse. The exposure is not an accident of a few unlucky TLDs; it is a structural consequence of two facts. First, the morph cost is a coding distance between short byte strings, and short strings that share most of their bytes are close codewords: the two-character TLDs occupy the fewest and most crowded codeword positions, so they collide the most, which is why length two carries 1881 one-module pairs while length four carries only 39. Second, the number of same-length TLDs is large and growing. The root once held a few hundred TLDs; the 2012 new-gTLD programme expanded it to the 1285 ASCII delegations (and 151 internationalised ones) we measure here, and every delegation adds a new node to the morph graph and new edges to its length class. Morph density scales roughly with the square of the number of same-length TLDs, so each expansion of the root increases the exposure super-linearly. Lengths five and above are progressively less exposed, for the same codeword-position reason, but the short classes that dominate real-world branding, the two- and three-character TLDs, are exactly the crowded ones. The exposure is therefore a durable and increasing property of the root zone, and it is best understood and defended at the level of the root, not of any single registrant.

5.3 *The dangerous subset: registrable open-to-open morphs*

The big counts are not the operational surface. Most one-module two-character pairs are between obscure ccTLDs no brand uses (for example `.cg` to `.cf`), and the cheapest morphs of a given host are often to TLDs nobody trusts: the one-module targets of `kaveh.org` are `.cbn` and `.msd`, which illustrate that "cheapest morph" and "credible attack" routinely diverge. The dangerous subset is narrower: morphs where the attacker can *own* the target, so a victim who scans the edited code arrives at an attacker-controlled site. We call that a *takeover*. Reaching a target the attacker cannot own is the weaker case: the edited code lands on a lookup failure or on the genuine government or brand host, which is misdirection, not a hijack.

Table 5 therefore leads with the registrable open-to-open morphs, ordered by added-ink cost. Of its 11 targets, 10 are open and one (`.aws`) is a single-registrant brand. Figure 10 shows three at print scale. A single added dot turns the open ccTLD `.ai` into `.am`, `.co` into `.cm`, and `.net` into `.new`; two dots turn `.com` into the open gTLD `.cam`. These added-ink costs are instance-specific (metric B): they are measured for the real label `amazon.*` at mask 0, the same labels the live explorer uses, and the added module can fall in a parity position rather than on the TLD glyph, since the attacker adds the cheapest dot anywhere.

PREPRINT

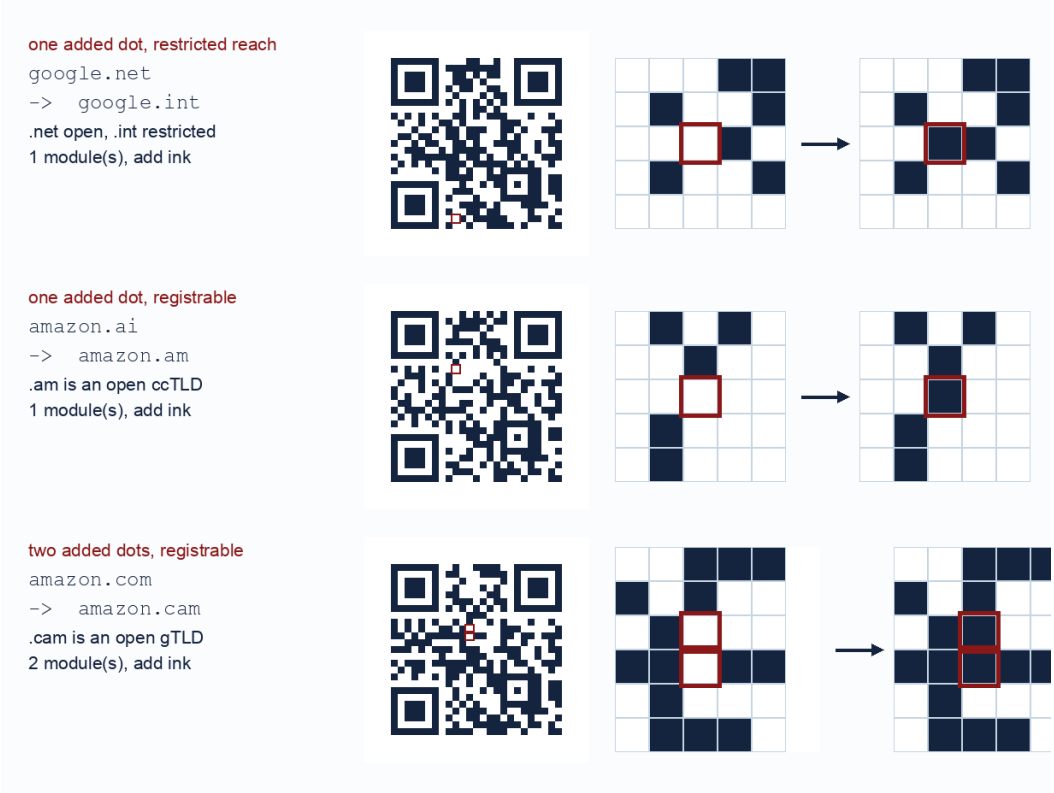


Figure 10. *Three same-length TLD morphs at print scale (label `amazon.*`, mask 0, add ink), each with a five-by-five inset of the changed region. Top: `.net` to `.int`, one added dot (a reach the attacker cannot own). Middle: `.ai` to the open ccTLD `.am`, one added dot (a registrable takeover). Bottom: `.com` to the open gTLD `.cam`, two added dots.*

From	To	Add ink	Target class	Effect
`.ai`	`.am`	1	open	takeover
`.io`	`.sg`	1	open	takeover
`.co`	`.cm`	1	open	takeover
`.net`	`.new`	1	open	takeover
`.com`	`.med`	1	open	takeover
`.ink`	`.inc`	1	open	takeover
`.live`	`.life`	1	open	takeover
`.com`	`.cam`	2	open	takeover
`.org`	`.dev`	2	open	takeover
`.dev`	`.net`	2	open	takeover
`.abc`	`.aws`	1	brand	no takeover

Table 5. *High-value morphs between real TLDs, open-to-open first (label `amazon.*`, mask 0, added-ink cost). All but the brand target `.aws` are openly registrable. The effect column records whether the attacker can register the target and take the name over.*

The classification (*open* / *brand* / *restricted*) is a committed, versioned artifact, `data/tld-classification.json` (version 2026-11-15), loaded and audited by the harness like every other number: the restricted set is the well-known sensitive, sponsored, and infrastructure labels (`.gov`, `.mil`, `.edu`, `.int`, `.arpa`, `.post`), and the brand set is a curated subset of ICANN Specification 13 single-registrant dot-brands. Its limits are stated in the file and real: the brand list is not exhaustive;

PREPRINT

registrable means a TLD is not categorically closed, not that a specific second-level label is free; and large brands register defensively across popular TLDs, so the open-morph counts are an upper bound on the attack surface, not the surface itself.

Table 6 is the contrast class: real TLDs the attacker cannot own but a victim trusts. `.net` reaches `.int` in 1 module and `.com` reaches `.gov` in 2 modules. There is no takeover here: the edited code lands on the genuine government or treaty-organisation host, or on a lookup failure, which is misdirection, weaker than owning the name. (`.com` to `.mil` at 4 modules is listed for completeness but is neither one dot nor stealthy.)

From	To	Modules	Target class	Effect
`.net`	`.int`	1	restricted	no takeover
`.com`	`.gov`	2	restricted	no takeover
`.com`	`.mil`	4	restricted	no takeover

Table 6. *Restricted namespaces reached cheaply (any polarity, label length six, version 2 level M). The attacker cannot own the target, but the boundary symbol still reaches it.*

5.4 *Single-operation analysis*

Because adding ink is the realistic field move, we separate morphs with a pure add-ink path from those that need both operations. There are thousands of pure-add morphs within the length classes we measured, and many are a single dot: `.afl` to `.nfl`, `.abc` to `.aws`, `.aig` to `.ong`, each one added dot at a six-character label. The natural guess that `.com` reaches `.net` by adding a few dots is wrong in an instructive way: the morph is 6 modules and *mixed*, with no pure add-ink and no pure remove-ink path, so it needs both operations. By contrast `.com` to `.org` is 5 removed and `.org` to `.com` is 5 added: direction matters, and the same pair of TLDs is a pure-add morph in one direction and a pure-remove morph in the other.

5.5 *The look-alike regime: where coding and visual similarity meet*

QR-Inking does not rely on the names looking alike: the attack is in the coding channel, not the glyphs, so a morph can be dangerous even when A and B look nothing alike. That is a strength over homograph attacks, and it is also why, for same-length *ASCII* morphs, the coding channel and the visual channel are *orthogonal*. Our companion font-grounded confusables resource, which renders every domain-relevant codepoint through the actual fonts browsers paint into the URL bar, scores zero ASCII homoglyph pairs as look-alike in either the strict or the at-a-glance tier across all three browser font cells; the only pair that registers at all, the digit `1` versus the letter `l`, scores 0.47, below every cut, and even `org` to `ong` is not font-identical because `r` and `n` are distinguished. Cross-script look-alikes are multibyte, so they cannot be a same-length ASCII morph. One caveat, "at a glance": a single-character ASCII morph shares most of the string and may slip past a careless reader (`google.com` versus `google.cam`), but the changed characters are not font-confusable, so a reader who looks sees a different name. The genuine look-alike regime is IDN.

For internationalised TLDs the two channels combine. Among the 151 internationalised (`xn--`) TLDs there are 9 same-length pairs within three modules. Table 7 lists four, each decode-verified on both decoders, both endpoints real delegated IDN TLDs, classified by how the two displays relate. Two kinds matter, and they are not the same.

> *A simplified and a traditional Chinese top-level domain are the same word to a reader. Two inked dots move a QR from one to the other, and the preview shows the same word.*

PREPRINT

The genuine *visual* look-alikes are the simplified-versus-traditional CJK ccTLDs: a Chinese ccTLD in simplified form becomes its traditional-character variant in two modules, and the two read as the same word (Figure 11). These are RZ-LGR variants, which a registry may bundle to a single registrant; where it does, the morph lands on the same operator, so there is no takeover, but the visual-channel defeat holds either way, because the preview shows the same word. The *same-meaning* case is different: the Bharat (India) family is one meaning across Odia, Bengali, Kannada, Gujarati, Devanagari, and Gurmukhi, all delegated and all a cheap morph apart. It is a conflated-root linguistic equivalence, not a visual look-alike, and we label it so. (Separately, `.xn--j1aef`, the Cyrillic *kom*, already imitates `.com`, though its cheap morph target is not itself a look-alike, so it is not in the table.) This joins QR-Inking to the font-grounded confusables work (the visual axis) and to the conflated-root-zone variant-TLD problem (ICANN RZ-LGR, the linguistic axis).

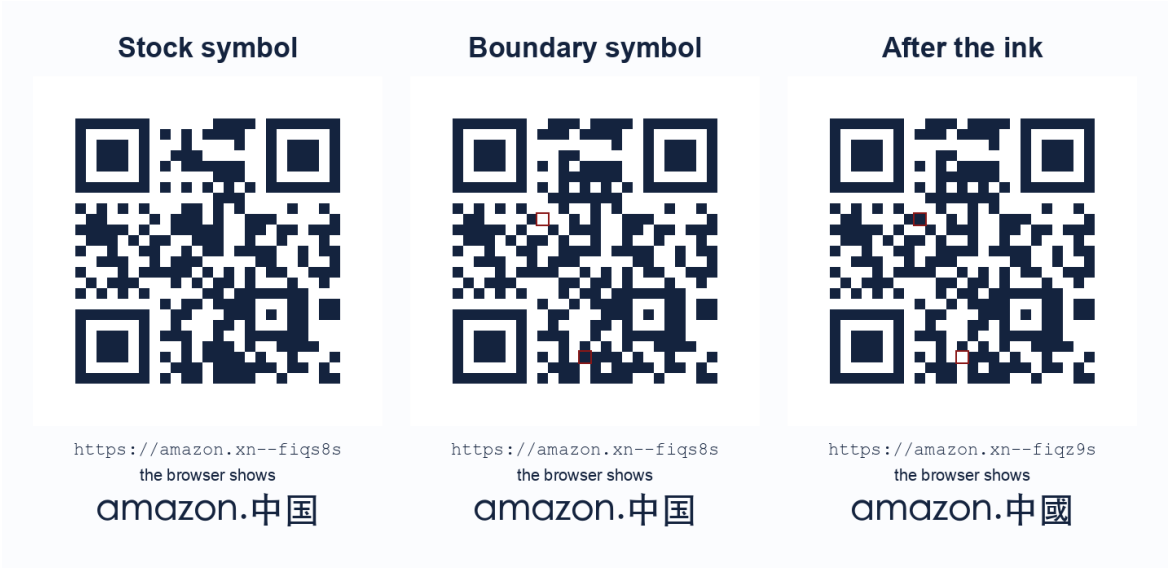


Figure 11. *An IDN look-alike morph on real delegated TLDs. The A-labels on the wire (`amazon.xn--fiqs8s` and `amazon.xn--fiqz9s`) differ; the browser shows the simplified and traditional character forms of the same word (China). Two modules (module distance), decode-verified on OpenCV and zxing-cpp, and flipped on a phone in the field (Section 6). At figure size the glyphs differ; at the small size of a URL preview the two read as the same word.*

From	To	Mod.	Kind	Effect	Displays as
`.xn--fiqs8s`	`.xn--fiqz9s`	2	visual-identical	no takeover	China, simplified and traditional
`.xn--kprw13d`	`.xn--kpry57d`	2	visual-identical	no takeover	Taiwan, simplified and traditional
`.xn--czrs0t`	`.xn--czru2d`	2	same script	takeover	store and mall (Chinese)
`.xn--3hcrj9c`	`.xn--45brj9c`	2	same meaning	no takeover	India (Bharat), Odia and Bengali script

Table 7. *Same-length IDN-TLD look-alike morphs (label `amazon.*`), both endpoints real delegated TLDs, each decode-verified on two independent decoders. The A-label is encoded on the wire; the browser displays the U-label.*

5.6 *Real-world targets and damage*

Content-independence (Proposition 1) turns the exposure map into a target list: the module cost is a property of the second-level label length and the TLD pair, not the brand text, so it carries unchanged from the `amazon.*` sweep to any real name. The target that matters most is the one carried by the printed-QR payment economy of Section 1.

Figure 12 is that case. `qr.alipay.com` followed by a short code is Alipay's real merchant-presented QR, the host a shopper's camera resolves at a counter across China [22]. At version 3, level M, a single directed inked module (add one dot) turns it into `qr.alipay.cam`, decode-verified on both decoders: the stock and boundary symbols read `qr.alipay.com`, the edited symbol reads `qr.alipay.cam`. One dot moves the host.



Figure 12. *Alipay's merchant-presented QR, `qr.alipay.com` to `qr.alipay.cam`, version 3 level M (the merchant path pushes the URL past version 2 capacity). Stock and boundary both decode `qr.alipay.com`; one added module (ringed, magnified below) decodes `qr.alipay.cam`. The module distance is 1 here, and it is a function of the version, the TLD's byte position, and the TLD pair. Proposition 1 fixes only the brand-text independence: at a fixed version the cost does not depend on the label text. Table 8 gives the clean same-version comparison, `venmo.com` to `venmo.cam` at one module and `alipay.com` to `alipay.cam` at two, both version 2, differing only in label length. The path shown is a representative merchant code.*

The morph is not academic, which is visible in who already owns the neighbour. On `.cam`, a verified-open gTLD, we read the registration of the one-edit `.cam` neighbours of major payment and account hosts from RDAP (which exposes the registrar, not the redacted registrant). Among the registered ones, two patterns appear (Table 8). Some sit at the brand's own corporate registrar, a defensive registration by the same owner: `alipay.cam` at Dominet, Alibaba's registrar (IANA 3775) [23], and `venmo.cam` at MarkMonitor, PayPal's. Others sit at a registrar the brand does not use, held by a party we can identify only as not the brand: `paypal.cam` (Dynadot), `stripe.cam` (joker.com), `ledger.cam` (Spaceship), and `weixin.cam` (Gname).

The pattern that matters most is the one Table 8 does not print. Some one-edit `.cam` neighbours of major payment brands are *unregistered right now.* That is the open flank: the attacker breaks nothing, because the look-alike is already sitting one ink edit from a live payment host, and the

name can be registered today for a few dollars. Operators are buying some neighbours defensively, strangers hold others, and the ones that would hurt most are still unclaimed. We notified the affected operators before this release, and we do not name the unregistered neighbours here.

Printed host	One-edit morph	Mod.	Registration (RDAP)	Damage if scanned
`alipay.com`	`alipay.cam`	2	held, same owner	payment redirect (Alipay)
`venmo.com`	`venmo.cam`	1	held, same owner	P2P payment redirect
`weixin.com`	`weixin.cam`	2	held, third party	login and pay (WeChat)
`paypal.com`	`paypal.cam`	2	held, third party	payment and login capture
`stripe.com`	`stripe.cam`	2	held, third party	merchant payment page
`ledger.com`	`ledger.cam`	2	held, third party	wallet seed phrase (irreversible)

Table 8. *Real payment and account hosts one* `.cam` *edit from a look-alike, restricted to neighbours that are already registered. Module counts are engine-computed, any-polarity, each at the smallest QR version its own URL needs (level M); the count is the engine's per-URL output, not a length rule. Registration status is from RDAP on 2026-11-16:* same owner *is inferred from the brand's own corporate registrar,* third party *from a registrar the brand does not use. Unregistered neighbours (the open flank) are real but are not printed here. As throughout, an open target means the name can be registered, not that any specific morph is registered or live, and the registrar is a fact while the owner behind it is an inference.*

The sharpest case also defeats a reader who checks the preview. In the look-alike regime (Section 5.5), a Chinese-language store-to-mall ccTLD pair is a registrable two-module morph, and the simplified and traditional forms of China's own ccTLD are two modules apart and read as the same word, which bites wherever a camera or browser surfaces a URL to read. Many codes in these flows resolve inside a super-app rather than a browser, with no URL preview at all; there an ASCII morph has nothing for an attentive reader to catch, so the preview defense never runs. Either way, this regime meets the largest printed-QR surface in the world.

6. Validation

Every number in this paper is produced by one committed harness, `src/validate.py`, which writes a machine-readable report to `out/validation.json`, a human table to `out/validation.txt`, the figures, and the LaTeX macros and table bodies this document reads. In this build it runs 166 checks and 166 pass. The tests are decoder tests on the module matrix and syndrome tests on the codewords.

The encoder, and what it covers. The matrix this code writes for a URL, at a chosen version, level, and mask with no preload, is compared module for module against segno 1.6.6. Across 120 comparisons (every configuration at all eight masks, including 5 multi-block layouts, Table 9) the maximum module difference is 0, and the byte-mode payload read back from the data codewords equals the input URL every time. This is not an independent reimplementation of the encoder: our `render()` reuses segno's own `make_blocks` (Reed-Solomon), `add_codewords` (zigzag placement), `find_and_apply_best_mask`, and `add_format_info`/`add_version_info`. What the comparison validates is that our codeword assembly and our ISO 7.6 interleaving reproduce segno's module output; a bug shared with segno would not show. The genuinely independent, end-to-end check of the attack is the decode, below.

PREPRINT

Config	Blocks	Masks	Max module diff
2-L `google.com`	1	8	0
2-M `google.com`	1	8	0
2-Q `google.com`	1	8	0
3-L `example.com`	1	8	0
3-M `example.com`	1	8	0
3-Q `example.com`	2	8	0
3-H `example.com`	2	8	0
4-L `google.com`	1	8	0
4-M `google.com`	2	8	0
5-L `google.com`	1	8	0
5-Q `google.com`	4	8	0
6-M `google.com`	4	8	0
1-L `a.co`	1	8	0
1-M `a.co`	1	8	0
1-L `apple.com`	1	8	0

Table 9. *Encoder agreement with segno 1.6.6. Each configuration is compared at all eight masks; the maximum module difference is the worst case over those masks.*

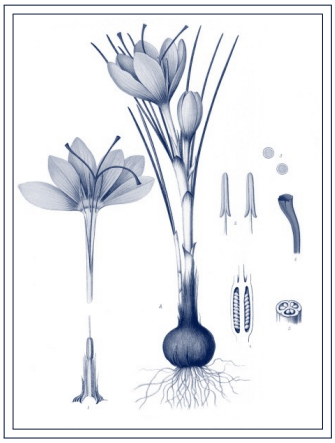
Two independent decoders. For every lead pair, four decodes: the stock symbol decodes to A, the boundary symbol decodes to A, the boundary with the user modules edited decodes to B, and the stock symbol with only those modules edited still decodes to A (the negative control). Every decode is run on two independent decoders, OpenCV 5.0.0's Aruco detector and zxing-cpp, and they agree on all 11 of 11 host leads and all 15 of 15 TLD morphs. Table 10 is the TLD decode set, 15 of 15 representative morphs, spanning legacy gTLDs, high-value gTLDs, two-character ccTLDs, and the `kaveh.org` and `google.com` examples; both endpoints of every row are real delegated TLDs. The IDN look-alike morphs of Section 5.5 pass the same two-decoder test, 4 of 4.

From	To	Ver.	Mod.	Decoded
`google.com`	`google.cam`	2-M	2	yes
`google.com`	`google.gov`	2-M	2	yes
`google.net`	`google.int`	2-M	1	yes
`google.com`	`google.net`	2-M	6	yes
`google.org`	`google.com`	2-M	5	yes
`kaveh.org`	`kaveh.ong`	2-M	2	yes
`amazon.bar`	`amazon.car`	2-M	1	yes
`amazon.aco`	`amazon.eco`	2-M	1	yes
`amazon.aig`	`amazon.ong`	2-M	1	yes
`amazon.afl`	`amazon.nfl`	2-M	1	yes
`amazon.cc`	`amazon.cf`	2-M	1	yes
`amazon.cm`	`amazon.co`	2-M	1	yes
`amazon.care`	`amazon.case`	2-M	1	yes
`amazon.band`	`amazon.land`	2-M	1	yes
`google.abc`	`google.aws`	2-M	1	yes

Table 10. *TLD decode set. For each morph the stock and boundary symbols decode to the source and the edited symbol decodes to the target, on both OpenCV and zxing-cpp. Both endpoints are real TLDs.*

PREPRINT

Physical field demonstration. The attack is a pen stroke on paper read by a phone, so we ran one. On an iPhone 14 Pro Max (iOS 27.2, the native Camera app's VisionKit QR detector, a third independent decoder beyond OpenCV and zxing-cpp), a printed boundary symbol decoded as A; a hand-inked edit with a marker, adding ink only, turned it into B, which the phone decoded. Two examples flipped: `google.com` to `google.cam` with two inked squares, and the simplified-to-traditional Chinese IDN morph of Section 5.5 with three inked squares (the three-versus-two gap is exactly the two metrics: the module distance is two, the add-ink cost is three). The one-module construction leaves zero error-correction margin, yet a clean print and a careful fill flipped reliably. This is a successful field demonstration and an existence proof of real-world viability, not a controlled study: one device, one tester, two examples. Robustness across printers, sizes, lighting, and devices, and an abandonment study, remain future work.



Crocus sativus, saffron. Köhler plate 194. The most adulterated spice in the trade, bulked with safflower and marigold florets dyed to the same red; by eye the fake is the genuine article, and only a laboratory test tells them apart. Validation here is that test, run on two independent decoders and a phone.

Zero margin. The construction is fragile by design, and this cuts two ways. A one-module boundary sits at distance exactly t from A with no residual budget toward B, so any additional print or capture defect can push it past the radius (it fails to read A) or across it (it tips to B). Field reliability therefore depends on print and scan quality in a way an ordinary code does not, which is a limit on the attack; and *benign* damage to a boundary print can accidentally retarget the host, which is a safety and forensics point. We state the casual-scan survivability at the level we actually demonstrated: the coding distances, agreement of two software decoders and one phone decoder, and the two field flips above, not a field reliability rate.

Polarity, placement, self-checked decoder. Each user module prints the requested direction and flips the right way (add ink is 0 then 1, remove ink is 1 then 0); a plan that would flip a module the wrong way is rejected, and the character under each host-case module is recovered from the bit placement and matched to the character the edit changed. The detection results use a bounded-distance Reed-Solomon decoder written from scratch over $\mathrm{GF}(2^8)$; the harness confirms it reports zero corrections on a valid segno codeword, exactly k corrections for randomly sampled patterns of each weight $k = 1, \ldots, t$, and *uncorrectable* (not a miscorrection) for sampled weight-$(t{+}1)$ patterns, which is the property the detection section relies on.

7. Detection and mitigation

Detection follows from the construction rather than from a signature. The boundary print decodes to A and already contains t corrected symbols, concentrated on one neighbour's difference set. Two cheap tests separate it from ordinary print damage.

Correction count. A decoder that reports how many symbols it corrected can flag a scan that is already sitting on the radius. Figure 13 is the trace from the self-checked decoder on the version-2 level-M example: a clean scan corrects 0 symbols, a boundary scan corrects 8, and a sensible rule flags any scan whose correction count is t or $t-1$. The finer signal is that the corrected positions of a boundary print are exactly the symbols where A and one candidate neighbour disagree, which ordinary damage does not reproduce.

PREPRINT

Single-module test. Inverting each data module in turn and recomputing the codeword distance separates the one-module boundary from a stock symbol: on a stock `google.com` symbol 0 single-module inversions bring the word within radius t of the neighbour, while on the boundary symbol exactly 1 does. The reported figures are codeword-distance counts (whether the inverted word is within t of B and beyond t of A), not repeated runs of an image decoder; format information gives the level, so t is known without guessing.

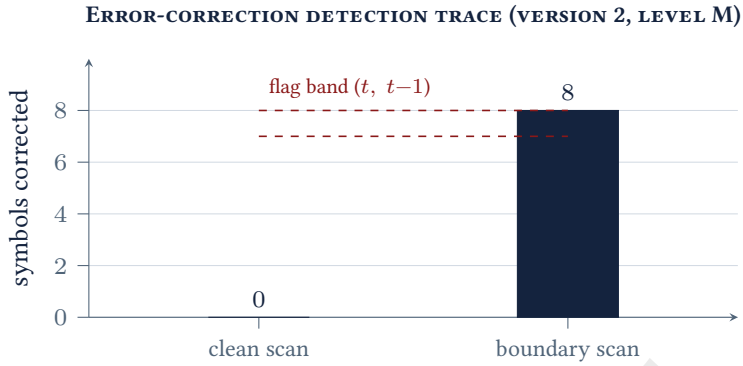


Figure 13. *Symbols corrected by the self-checked Reed-Solomon decoder, clean scan versus boundary scan, version 2 level M. A clean scan corrects 0; the boundary scan corrects 8, on the radius. The dashed band is the suggested flag threshold at t and $t-1$. The single-module test (prose) is a codeword-distance count: 0 inversions retarget a stock symbol, 1 the boundary symbol.*



Myristica fragrans, nutmeg. Köhler plate 97. Pedlars once whittled nutmegs from wood that passed for the real seed until someone tried to grate one, so that a wooden nutmeg became an old word for a fraud. The grater, here, is a decoder that reports how many symbols it corrected: a boundary print cannot hide that it is already on the radius.

For completeness, the image detectors agree with the codeword model: OpenCV's Aruco detector accepted all 20 random preloads of the `googlg` pair and all 361 of 361 deterministic plans in the version-2 level-M sweep; the legacy `QRCodeDetector` accepted 359 of 361, its two misses (`gfogle.com, gougle.com`) being two-module plans at the edge of its radius that a different preload recovers.

Mitigation. Three mitigations follow. Scanners and scanning libraries can expose the correction count and warn when a clean scan is already near the radius. High-assurance print channels (payment stands, official notices) can raise the error-correction level deliberately to an odd n_{ec}, which forces the cheapest edit to two modules (Section 7.1), or reject symbols whose correction count is non-zero at acceptance scanning. Registries and brand-protection programmes can treat the same-length TLD neighbours of a defended name as part of the brand's attack surface, exactly as typosquats are treated today, using the exposure map of Section 5.

7.1 *Limits*

Odd n_{ec}. When the parity count is odd, $d_{min} = 2t+2$, so a minimum-weight difference codeword ($|D| = d_{min}$) gives $|U| = |D| - 2t = 2$ and the cheapest edit is two modules before any direction filter. Version 3 level L is the case for `google.com` (2 modules, `google.som`), and version 1 level L is why the natural encoding of `kaveh.org` needs two modules. An odd-n_{ec} level is a mitigation, not a vulnerability.

PREPRINT

More than one block. From version 3 level Q upward the message is several interleaved blocks, and a character or TLD that straddles a block boundary pays one user symbol per block it touches, so the cheapest edit can be two modules even at minimum weight.

Decoder radius. ISO/IEC 18004 clause 7.5.1 lets an implementation hold p syndromes back for misdecode protection, but only for the small configurations where $p > 0$ (versions 1-L/M/Q/H, 2-L, 3-L). The version-2 level-M headline has $p = 0$, so every decoder agrees there, and both of our software decoders plus the field phone correct to the full radius $\lfloor n_{\text{ec}}/2 \rfloor$ in practice: the one-module result is not decoder-specific. Where $p > 0$ it matters: on version 2 level L a strict reading corrects 4 errors, not 5, and the cheapest `google.com` edit grows from 1 to 3 modules. A deployment that cannot name its decoder should assume the smaller radius.

Length-changing edits and witnesses. Changing the length of the string, including a TLD of a different length, moves the character-count header and is a different distance that we do not rank. The host-typo targets are coding witnesses, not claims about registration or visual confusability. The TLD targets are different: both endpoints are real delegated TLDs, which is the entire point of Section 5.

8. Related work

QR structure abuse. The boundary geometry is the deterministic endpoint of the fake-QR construction of Takita, Okuma, and Morii [2], specialised to the hostname and TLD bytes and to a one-direction edit. Ohigashi, Kawaguchi, Kobayashi, and colleagues [3] detect that family from the error-correction trace, and the same trace catches the boundary symbol, because a clean scan is already on the radius. Our contribution over this line is the directed (single-operation) edit, the content-independence result, and the IANA-root measurement.

PREPRINT

QR phishing (quishing) and the URL preview. The dominant real-world technique is stickering a fully-malicious QR over a legitimate one [9,15], which needs no coding theory; QR-Inking's only edge is the boundary property (Section 2). The human premise, that a scanned code is acted on with little scrutiny, is measured by the modern field studies we engage directly: Sharevski et al.'s 2022 field study [14] found a majority will proceed, and the 2024 naturalistic study [15] found 67% of scanners opened the link without inspecting the URL. But that same study is the reason we do not overclaim: abandonment rose from 8% when the preview looked legitimate to 44% when the domain visibly contained "phish," so a *visible* discrepancy is exactly where preview-readers bail. Since our ASCII morphs are deliberately non-confusable, a reader who inspects the preview sees a different name; our answer is the IDN look-alike subset (Section 5.5), where the preview shows what reads as the same word, confirmed on a phone in the field (Section 6).



Cinchona bark once more. The fight to tell genuine febrifuge from cheap filler has a literature centuries older than ours, and the homograph and quishing studies below carry that literature into names and printed codes.

Physical QR tampering. The surface we exploit is catalogued in the usable-security literature: Krombholz and colleagues [16] survey QR attacks including module-level tampering, and Dabrowski and colleagues [17] demonstrate physical barcode manipulation, a barcode printed inside a barcode. The FBI's IC3 warned in 2022 that criminals tamper with QR codes, physically and digitally, to redirect payments [18]. What the field lacks is measurement: the

published corpus is clean synthetic QR images, and there is essentially no public dataset of physical tampering. QR-Inking's reproducible harness, with its two-decoder and phone validation on real prints, is a step toward that missing ground truth.

Homograph and confusable attacks. The domain-confusability literature, from the homograph attack of Gabrilovich and Gontmakher [6] through the measurement studies of Holgers et al. [7], the reexamination of IDN homographs by Liu et al. [8], and the Unicode security mechanisms of UTS #39 [10], concerns how two names look alike to a reader. In the modern scan pipeline the decoded URL *is* surfaced as text, so for ASCII morphs QR-Inking is subject to the same reader-scrutiny defence this literature studies, and is worse off because it declines to make the target look alike; the IDN regime is where it is subject to, and defeats, that defence. This paper is a companion to our font-grounded confusables work on the visual channel.

Root-zone string similarity. ICANN's New gTLD String Similarity Review [13] already evaluates root strings for confusability, but it is "solely concerned with visual confusability" (and, in the next round, aural and meaning similarity). Module distance is an *orthogonal* similarity metric the review does not check: two visually, aurally, and semantically distinct TLDs can be one printed module apart. Coding distance is a blind spot of a review built for the eye.

Coding theory and standards. The construction rests on the maximum-distance-separable property of Reed-Solomon codes [4], the Berlekamp-Massey error-locator recovery [5] used by our self-checked decoder, and the QR symbology of ISO/IEC 18004 [1].

9. Scope and ethics

The material here is derived entirely from public data: the IANA root zone TLD list (Version 2026111500) and the public QR and Reed-Solomon standards. Both endpoints of every morph we display are already-delegated TLDs, so the paper describes a structural property of the root zone and the QR encoding, not a vulnerability in any one operator's system. The host-typo targets are coding witnesses, not claims that a particular second-level name is registered or visually confusable. We ship the detection test (Section 7) and the mitigations alongside the construction, and the candidate generator is pluggable and brand-agnostic: it enumerates coding distance and leaves the choice of interesting targets to whoever runs it.

Where our real-world scan (Section 5.6) found an unregistered one-edit neighbour of a live payment host, we notified the affected operator before this release and do not print the grabbable names.

10. Conclusion

A printed QR code is a trust channel made of ink, and ink can be edited. We showed that an attacker who prints a boundary symbol can retarget the decoded host with a single directed module edit, that the edit can be constrained to adding one dot, and that an ordinary QR code is nonetheless not one dot from a lookalike.

The result that matters for the DNS is structural. For a given same-length TLD pair, the module distance of the morph depends only on the length of the second-level label (and is mask-independent), so the exposure is a property of the root zone, and the root zone is densely exposed. Among two-character TLDs alone there are 1881 one-module morph pairs.

The operationally dangerous ones are the registrable open-to-open morphs, where the attacker can take the name over: a single added dot takes `.ai` to `.am` or `.net` to `.new`. Reaches into `.gov`, `.int`, and `.mil`, which the attacker cannot own, are a lesser harm: misdirection, not takeover.

PREPRINT

On real payment brands this is already live. The one-edit `.cam` neighbours of major payment hosts are being registered defensively by the brands themselves, held by unrelated third parties, or left unregistered and open for anyone to take, and that last group is the flank that should worry an operator most.

The one regime where the attack also defeats a URL-reading user is internationalised TLDs, where a simplified-form Chinese ccTLD becomes its traditional variant in two modules and, at URL-bar size, the preview reads as the same word, which we confirmed on a phone. The construction, the measurement, the two-decoder validation, and the detector are all regenerated by a committed harness.



Cinnamomum verum once more, to close where the cover began. The whole paper is a long gloss on this plate: a thing that passes for another until someone runs the test. The printed code is the cassia, and the harness is the chemistry.

PREPRINT

References

[1] ISO/IEC 18004:2015. *Information technology, automatic identification and data capture techniques, QR Code bar code symbology specification.* Block structure, generator, mask patterns, and format information as implemented by segno 1.6.6. (A fourth edition, ISO/IEC 18004:2024, now supersedes the 2015 third edition; the clauses cited here are unchanged, and the measurements target segno's 2015(E) conformance, so the 2015 reference is retained.)

[2] M. Takita, H. Okuma, M. Morii. *A Construction of Fake QR Codes Based on Error-Correcting Codes.* CANDAR 2018, pages 188 to 193. DOI 10.1109/CANDAR.2018.00033.

[3] T. Ohigashi, S. Kawaguchi, K. Kobayashi, et al. *Detecting Fake QR Codes Using Information from Error-Correction.* Journal of Information Processing, vol. 29, pages 548 to 558, 2021. DOI 10.2197/ipsjjip.29.548.

[4] I. S. Reed, G. Solomon. *Polynomial Codes over Certain Finite Fields.* Journal of the Society for Industrial and Applied Mathematics, vol. 8(2), pages 300 to 304, 1960. DOI 10.1137/0108018.

[5] E. R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill, 1968; with J. L. Massey, *Shift-register synthesis and BCH decoding*, IEEE Transactions on Information Theory, vol. 15(1), pages 122 to 127, 1969, DOI 10.1109/TIT.1969.1054260. The Berlekamp-Massey error-locator recovery used by the harness's self-checked decoder.

[6] E. Gabrilovich, A. Gontmakher. *The Homograph Attack.* Communications of the ACM, vol. 45(2), page 128, 2002. DOI 10.1145/503124.503156.

[7] T. Holgers, D. E. Watson, S. D. Gribble. *Cutting through the Confusion: A Measurement Study of Homograph Attacks.* USENIX ATC, 2006, pages 261 to 266.

[8] B. Liu, C. Lu, Z. Li, et al. *A Reexamination of Internationalized Domain Names: The Good, the Bad and the Ugly.* IEEE/IFIP DSN 2018, pages 654 to 665. DOI 10.1109/DSN.2018.00072.

[9] T. Vidas, E. Owusu, S. Wang, et al. *QRishing: The Susceptibility of Smartphone Users to QR Code Phishing Attacks.* Financial Cryptography and Data Security (USEC) 2013, LNCS 7862, pages 52 to 69. DOI 10.1007/978-3-642-41320-9_4.

[10] Unicode Consortium. *Unicode Technical Standard #39 (UTS #39): Unicode Security Mechanisms.*

[11] IANA. *Root Zone Database and the list of top-level domains*, `tlds-alpha-by-domain.txt`, Version 2026111500.

[12] segno 1.6.6 (reference encoder); OpenCV 5.0.0 `QRCodeDetectorAruco` and zxing-cpp (the two independent reference decoders).

[13] ICANN. *gTLD Applicant Guidebook* (2012-06-04), Module 2, §2.2.1.1 String Similarity Review; and *String Similarity Review Guidelines* (Public Review Draft), 7 February 2024, which is "solely concerned with visual confusability."

[14] F. Sharevski, A. Devine, E. Pieroni, P. Jachim. *Phishing with Malicious QR Codes.* EuroUSEC 2022, pages 160 to 171. DOI 10.1145/3549015.3554172. (Preprint: *Gone Quishing*, arXiv:2204.04086.)

[15] F. Sharevski, M. Mossano, M. Veit, G. Schiefer, M. Volkamer. *Exploring Phishing Threats through QR Codes in Naturalistic Settings.* USEC 2024. DOI 10.14722/usec.2024.23050.

[16] K. Krombholz, P. Fruehwirt, P. Kieseberg, I. Kapsalis, M. Huber, E. Weippl. *QR Code Security: A Survey of Attacks and Challenges for Usable Security.* Human Aspects of Information Security, Privacy, and Trust (HAS) 2014, LNCS 8533, pages 79 to 90. DOI 10.1007/978-3-319-07620-1_8.

[17] A. Dabrowski, K. Krombholz, J. Ullrich, E. Weippl. *QR Inception: Barcode-in-Barcode Attacks.* ACM SPSM 2014 (Security and Privacy in Smartphones and Mobile Devices). DOI 10.1145/2666620.2666624.

[18] Federal Bureau of Investigation, Internet Crime Complaint Center (IC3). *Cybercriminals Tampering with QR Codes to Steal Victim Funds.* Public Service Announcement I-011822-PSA, 18 January 2022, `ic3.gov/PSA/2022/PSA220118`.

[19] Consultative Group to Assist the Poor (CGAP). *China: A Digital Payments Revolution.* September 2019.

[20] Press reporting on merchant QR-code swap fraud: CGTN, *Scan no scam: police arrest two fraudsters for cheating via fake QR codes*; South China Morning Post, *The rise of QR code scams in China* (2017); Nikkei Asia, *QR code scams strike China, from merchants to traffic tickets.*

[21] BIS Innovation Hub. *Project Nexus: enabling instant cross-border payments.* Bank for International Settlements, blueprint, 2024.

[22] Alipay (Ant Group). *Merchant-presented QR and order-string format*, developer documentation (the `qr.alipay.com` merchant-code URL).

[23] IANA. *Registrar record for Dominet (HK) Limited*, registrar IANA ID 3775 (the registrar of record for the `alipay.cam` registration read from RDAP).

Reproduce: `.venv/bin/python src/validate.py` regenerates `out/validation.json`, `out/validation.txt`, the figures, and the macros and tables this paper reads (166/166 checks pass). Research by Kaveh Ranjbar, Whisper Security. The paper is licensed CC BY 4.0; the reference code is MIT. Botanical plates from Köhler's Medizinal-Pflanzen (1887), public domain, cyanotype-toned.

PREPRINT