**Source:** cppcc docs/A_Second_Kind_of_Secret_Draft_0.1.md @ 0aa6460 — curated 2026-08-07; edits land in the source first.

# A Second Kind of Secret

## Structural hardness as a complement to cryptographic confidentiality

**Draft 0.1 — 2026-08-07**

*A short position paper for cryptographers and security researchers. It argues that a body of hardness results outside the usual cryptographic toolkit — grammatical inference — describes a secret with a property no key has; sets out precisely what would be required to make that cryptographically meaningful; identifies a primitive-design problem with no standard answer (a keyed, validity-preserving transformation on the structures of a context-free language); and points at a working implementation where all of it can be built and measured.*

*It is an invitation, not a claim of a result.*

---

## 1. What Kerckhoffs's principle settles, and what it assumes

Kerckhoffs's second principle is the field's founding commitment: a system must remain secure even if everything about it except the key is public knowledge. It was a corrective to security through obscurity and it has been vindicated for a hundred and forty years. Nothing below disputes it.

But it is worth separating what the principle *settles* from what it *assumes*. It settles where secrecy may not live: not in the algorithm, not in the protocol, not in the implementation. What it assumes — quietly, and so universally that the assumption is nearly invisible — is that the secret which remains is a **value**.

That assumption has consequences the field has spent its entire history engineering around. A value is an object in the world. It can be copied without being removed, which makes theft silent. It can be disclosed by a person, which makes the human the weakest component. It exists in memory, in a configuration file, in a backup, in a core dump, in a process environment, in a screenshot. Key rotation, escrow, hardware security modules, forward secrecy, split knowledge,

threshold schemes, key ceremonies — this entire apparatus is engineering around the fact that the secret is a thing which exists and can therefore be obtained.

A single disclosure is total and instantaneous. Every artifact ever produced under that key opens at once.

This is not a flaw in Kerckhoffs's principle. It is a property of choosing a value as the locus of secrecy. The question this paper raises is whether it is the only available choice.

## 2. A second kind of secret

Consider a data format whose interpretation is defined by a formal grammar — not a published standard, but a proprietary specification held by its author.

An artifact in that format is not encrypted. Its bytes are entirely exposed. And yet a reader who does not hold the grammar is not in the position of a cryptanalyst holding ciphertext. They are in the position of a linguist holding an undeciphered corpus. Their task is not to search a keyspace; it is to **infer a formal language from examples of its output**.

The distinguishing property is structural, and it is the reason this is worth a cryptographer's attention:

> **There is no key.** The secret is a specification, and a specification is not obtainable by copying — only by re-derivation.

Nothing can be exfiltrated from memory, because the secret is not in memory as a discrete object. Nothing can be leaked by an insider in the way a key is leaked — the equivalent act is handing over a document, which is a different and more detectable thing. There is no rotation problem, because there is nothing whose compromise the rotation would remedy. Compromise, when it happens, is per-format and requires intellectual work rather than access.

The obvious objection — *this is security through obscurity, which Kerckhoffs settled* — deserves a direct answer, and it is the reason §3 exists. Security through obscurity is objectionable because the difficulty of rediscovering a hidden design is **unquantified and usually small**. The claim here is not that the grammar is hidden. It is that recovering it is an instance of a problem class with a substantial and unfavourable complexity literature. Whether that literature is *strong enough* is precisely the open question, and §4 is unsparing about it.

## 3. What is known about inferring a grammar

Grammatical inference has been studied since the 1960s, largely outside the cryptographic literature. The results are consistently negative for the learner.

**Identification is impossible from positive examples alone.** Gold (1967) showed that no algorithm identifies an arbitrary language from any *superfinite* class — one containing all finite languages plus at least one infinite language — given only positive examples, however many. The regular languages are superfinite; so are the context-free languages. An adversary holding a corpus of valid artifacts, with nothing telling them what would be *invalid*, is in exactly this setting.

**Minimum consistent inference is NP-complete.** Given finite sets of accepted and rejected strings, deciding whether a DFA with at most $k$ states is consistent with them is NP-complete (Gold 1978; Angluin 1978).

**Approximation does not rescue it.** Pitt and Warmuth (1993) showed the minimum consistent DFA problem cannot be approximated within any polynomial factor unless P = NP. This closes the escape hatch that usually makes NP-hardness practically irrelevant.

**And — the result that should interest this audience most — learnability is already known to be *cryptographically* hard.** Kearns and Valiant (1994) proved that PAC-learning polynomial-size DFAs is as hard as inverting RSA, factoring Blum integers, or deciding quadratic residuosity. This is not an analogy but a reduction: **a polynomial-time PAC-learner for DFAs would break RSA.** The reduction runs from cracking the cryptosystem *to* the learning problem, placing learning at least as high as the standard hardness assumptions.

That result is the reason this paper exists. The bridge between the two fields was built thirty years ago and has been little travelled in this direction — as a *source* of hardness for a defensive construction, rather than as a limitation on machine learning.

For context-free grammars the picture is more severe still, since equivalence of two CFGs is undecidable.

## 4. What would be required, and what is missing

A cryptographer will already have identified the gap, and it should be stated plainly rather than discovered.

**Worst-case hardness is not average-case hardness.** The results of §3 say that *some* instances are hard. They say nothing about whether a particular grammar — one designed by an engineer for a real data format, with the regularities that human design tends to produce — is among them. A small, conventionally-structured grammar may well be recoverable in an afternoon by a competent reverse engineer who has never heard of Gold.

This is precisely the gap the field has crossed before. Lattice problems were known NP-hard long before they were cryptographically usable; what changed the situation was Ajtai's (1996) worst-case to average-case reduction, which made it possible to sample instances that are hard *because* the worst case is hard. **Grammar-based constructions have no equivalent.** No sampling procedure is known that produces grammars whose recovery is hard on average, and no reduction ties a random grammar's difficulty to a worst-case bound. Until such a thing exists, the material in §3 is suggestive rather than load-bearing, and the honest description of the present state is:

> a hardness *source* with a real literature, awaiting the constructions that would make it a hardness *assumption*.

**A membership oracle is the obvious attack, and the class boundary matters more than it first appears.** Angluin (1987) showed that regular languages are exactly learnable in polynomial time given a *minimally adequate teacher* — an oracle answering membership and equivalence queries. Any deployment exposing a parser as a queryable service supplies the membership half directly: submit input, observe whether it is accepted. Taken alone, that looks fatal.

Three things qualify it.

First, a real adversary rarely holds the *equivalence* oracle, which is the harder half of Angluin's construction — knowing when a hypothesis is correct generally requires already knowing the answer. In practice it is approximated by sampling, which weakens the result.

Second, and more decisively: **L* is a result about regular languages.** Angluin and Kharitonov (1995) showed that for context-free grammars — along with NFAs, two-way DFAs, and unions or intersections of DFAs — there is *no* polynomial-time prediction algorithm **even with membership queries**, assuming the intractability of quadratic residuosity, RSA inversion, or factoring Blum integers. Grammars of the kind discussed here are context-free. The oracle attack that defeats a regular-language construction therefore runs into a cryptographic barrier one class up, and it is the *same* barrier as §3's — the same assumptions, the same literature.

Third, that barrier is about *prediction*, and prediction is not what an attacker necessarily needs (see below). So the correct reading is not "oracles are harmless" but: the natural attack is answered at the right language class by a conditional result, and the residual worry is partial extraction rather than learning.

The design constraint stands regardless, because defence should not lean on a conditional result it does not have to: **a system relying on structural hardness should avoid behaving as an oracle at all.** Acceptance and rejection indistinguishable to the observer, uniform failures, no parser exposed as a service where that can be avoided. That is an engineering discipline with real answers, and it is where a cryptographer's instincts transfer immediately.

**Structure is not payload.** Grammar-derived hardness protects how an artifact is *organised*. A string literal inside it is readable by anyone who opens the file. This is not a limitation to be worked around — it is the reason the two mechanisms belong together.

**Exact recovery is not the adversary's goal.** The theorems concern recovering *the* grammar, or a minimal one. An attacker usually needs far less: enough approximate structure to locate one field. Hardness of exact inference is weak evidence about the cost of partial, targeted extraction, and no one has characterised the latter.

## 5. Why the two compose

The argument for stacking is not that structural hardness is strong. It is that it **fails differently**.

| | cryptographic key | structural secret |
|---|---|---|
| the secret is | a value | a specification |
| attack is | search | inference |
| obtained by | copying | re-derivation |
| insider risk | silent exfiltration | handing over a doc |
| compromise is | total, instant | partial, per-format |
| degrades against | compute | oracles, side info, public standards |
| maturity | average-case bounds, reductions | worst-case results, no reduction yet |

An adversary facing both must obtain a key *and* reconstruct a specification. Neither compromise implies the other, and the failure modes share no common cause: compute does not help against inference, and inference does not help against a cipher. Defence in depth is only meaningful when the layers are independent, and here the independence is structural rather than assumed.

What cannot yet be said is *how much* the second layer adds. There is no work factor, no reduction, no concrete bound. That is the honest position, and it is the invitation.

## 6. A transformation domain, not just a slot

Everything above treats the artifact as something to be *read* or not read. There is a second surface, and it is the one most likely to interest someone who designs primitives rather than deploys them.

A CCS artifact can be loaded back into a **Syntax-Controlled Runtime** — a structured, in-memory object rather than a byte string. Once it exists in that form, the interesting operation is not runtime-to-bytes but **runtime to runtime**: a transformation whose input and output are both valid structures over the same grammar.

Such a transformation is constrained in an unusual way. It must be

- **invertible**, exactly, so the original is recoverable;
- **validity-preserving**, so the output still parses; and
- **semantically faithful**, so what the artifact *means* survives.

A cipher over a byte range satisfies the first and ignores the other two, because it treats the artifact as an opaque buffer. A runtime-level transformation cannot: it operates on identifiers, rule numbering, symbol identity, ordering — objects with referential integrity to maintain.

### 6.1 The nearest neighbour, and where it diverges

Cryptographers have built in this direction before. **Format-preserving encryption** (Bellare, Ristenpart, Rogaway and Stegers 2009; NIST SP 800-38G, FF1 and FF3) encrypts while keeping the ciphertext in the plaintext's format — a card number stays a card number. FPE is the right mental model to arrive with, including its cautionary history: FF3 required revision to FF3-1 after cryptanalysis (Durak and Vaudenay 2017), which is a useful reminder that domain-preserving constructions are subtle rather than trivial.

The generalisation here is specific. FPE preserves a *format* — a regular, usually fixed-width domain. What is wanted here is preservation of **membership in the language of an arbitrary context-free grammar**. The domain is not a set of strings of known shape; it is defined by a specification, and it is the same specification whose recovery §3 argues is hard.

That gives the design problem a shape with no standard answer:

> Construct an invertible, keyed map on the structures of a context-free language that preserves validity and meaning, and measurably degrades what an inference algorithm can extract from a corpus of outputs.

### 6.2 Why this is where the open problem probably lives

§4 named the missing piece: no procedure is known for sampling grammars — or artifacts — whose recovery is hard on average. Notice what a runtime-level transformation actually does. Identifier renaming, symbol-identifier permutation, rule reordering, dead-rule elimination, controlled epsilon insertion: each leaves the language unchanged while removing exactly the human-meaningful regularities that make real grammars far easier to infer than the worst case suggests.

That is not a side benefit. **A keyed, validity-preserving transformation is a candidate hard-instance generator.** If an Ajtai-style reduction exists for this setting — open question 1, the one that decides whether any of this matters — it is difficult to see where else it would live. The transformation is the sampling procedure, and the key is what makes the sample unpredictable.

Whether any particular transformation family actually achieves that, and under what measure, is unknown. It is a well-posed question with a working substrate to test on, which is a better position than most open problems start from.

### 6.3 The stronger form: mapping between *different* grammars

Nothing requires the two sides of the transformation to share a grammar. The general form is a bijection

> **$T$ : structures over $G_1$ ⟷ structures over $G_2$**

where $G_2 \neq G_1$ and both directions are exact. What is required is not that the grammars match but that the mapping is **isomorphic** — every artifact has one image and one preimage.

This admits a construction with an interesting property. Let $G_2$ contain productions that have no counterpart in $G_1$: items the mapping knows about and discards on the return trip. They carry no information. To anyone holding only outputs they are ordinary structure, indistinguishable from the load-bearing kind, and there is nothing in an artifact that says which is which.

The consequence is worth stating carefully, because it changes what §3 is doing:

> **Inferring the observable grammar is no longer sufficient.** An adversary who completely succeeds at the §3 problem recovers $G_2$ — not $G_1$, and not T. The structural secret and the inference problem have become two independent unknowns rather than one.

That is a meaningful strengthening. §3's hardness results, and §4's honest caveats about them, describe the difficulty of the *first* unknown. Against a cross-grammar construction the adversary must clear that bar and then solve a second problem — recovering a keyed mapping — for which the first solution gives no purchase.

**Prior art, and the exact generalisation.** This is recognisably a relative of **format-transforming encryption** (Dyer, Coull, Ristenpart and Shrimpton, CCS 2013), which encrypts plaintext into ciphertext matching a chosen format, and was built to make one protocol's traffic identify as another's. FTE is the closest existing primitive and the right thing to compare against. Two differences define the space here: FTE targets formats specified by **regular expressions**, whereas this targets the structures of a **context-free grammar**; and FTE maps to a *string* in the target format, whereas this maps between *structured objects*, with referential integrity preserved on both sides.

The null-item idea is also a structural cousin of chaffing and winnowing (Rivest 1998), with the difference that the chaff here is grammatical rather than message-level, and is generated by the transformation rather than added alongside it.

⚠ **The failure mode this field already documented.** Systems that try to be unobservable *by imitating* another protocol have a poor record. Houmansadr, Brubaker and Shmatikov (2013) showed that SkypeMorph, StegoTorus and CensorSpoofer were all distinguishable by weak local observers, and concluded that unobservability by imitation is fundamentally flawed: matching a format is easy, matching a *distribution* — timing, error behaviour, the full range of real traffic — is not.

Whether that verdict applies here depends entirely on the goal, and the distinction is worth drawing sharply:

- If the goal is **confidentiality** — the adversary holds artifacts and must not read them — the Parrot critique is largely beside the point. Nothing needs to pass as anything; the artifacts are simply valid $G_2$ structures, produced by the same machinery that produces genuine ones. That is generation, not imitation, and it is the difference the critique turns on.
- If the goal is **unobservability** — the adversary must not be able to tell that CCS is in use at all — then Houmansadr's requirements apply in full, and a construction should be assumed distinguishable until someone demonstrates otherwise on distributional grounds.

The honest position is that the first goal looks defensible and the second should not be claimed.

## 7. What exists today

This is not a thought experiment. The Compiler Compiler System (CCS) produces artifacts of exactly the kind §2 describes: a compiled binary form whose interpretation is defined by the grammar that produced it. CCS is the subject of US Patent 8,464,232.

A working transformation tool — the CCS Obfuscator — is published under a source-available licence with its full technical compendium, including an unsparing account of the weaknesses of its own shipped cipher. It is deliberately built as a **platform**: a documented extension point where an algorithm is plugged in, with two oracles that check whether a transformation was honest —

- a **no-op round trip** proving the substrate is lossless on your data before you trust anything built on it;
- an **invertibility check** proving a transformation both changes the artifact and restores it exactly.

For a cryptographer the practical significance is the friction, or lack of it: implementing an algorithm here requires understanding an interface, not a compiler. The shipped cipher exists to be replaced.

Two transformation classes are specified but unimplemented, and they are the ones that bear directly on this paper: **identifier renaming** and **symbol-identifier permutation**. Both attack the inference problem rather than the payload, by removing the human-meaningful regularities that make a grammar guessable in practice — that is, by pushing an instance away from the easy average case and toward the hard worst case §4 says we cannot yet sample.

## 8. Open problems

Stated as questions, because that is what they are.

1. **Is there an Ajtai-style reduction here?** Can one sample grammars whose recovery is hard on average, with hardness tied to a worst-case bound? Without this, structural hardness stays suggestive.
2. **What is the actual work factor?** Given a grammar of stated size and a corpus of *n* artifacts, what does recovery cost? No published estimate exists.
3. **Do structural transformations measurably raise it?** Renaming and permutation should push instances toward the hard case. Should is not a measurement.
4. **How far does the Angluin–Kharitonov barrier actually reach?** It rules out polynomial-time *prediction* for CFGs even with membership queries, under standard assumptions. Does that transfer to the practical attack — partial, targeted extraction — or is there a gap between "cannot predict the language" and "cannot read one field"?
5. **Can a deployment credibly deny the oracle?** Uniform failure modes, indistinguishable rejection, non-exposed parsers — what does a rigorous version of that discipline look like?
6. **What does composition buy?** Is there a security notion under which cipher-plus-structure is provably stronger than either, or is the gain purely operational?
7. **Is there a keyed, validity-preserving transformation family on context-free structures with a defensible security notion?** §6's design problem, stated as the primitive it would be. FPE is the nearest prior art and does not reach this domain.
8. **Can such a family serve as the hard-instance generator §4 wants?** If question 1 has an answer, §6.2 argues this is where it lives.
9. **What does the cross-grammar construction of §6.3 actually buy?** It appears to add a second independent unknown, so that solving the inference problem yields the wrong grammar. Is that separation provable, or can the two be attacked jointly?
10. **How much null structure is useful, and does it show?** Grammatical chaff costs size and may itself be a statistical signature. Is there an optimum, and is it detectable?

Question 1 is the one that decides whether the rest matters.

## 9. What is being proposed

Nothing is being asserted here that requires belief. The hardness results are standard and citable; the limits in §4 are stated in the strongest form we know how to state them; the implementation is

public and its shipped cipher is documented as weak.

What is on offer is a **substrate and a seam**: a real artifact format with the structural property of §2, a working platform where an algorithm can be plugged in, and a set of questions that a specialist in this field is far better equipped to answer than its authors.

If any of §7 strikes you as tractable — or as already answered in literature we have missed, which would itself be valuable — we would like to hear from you.

**support@b3u.dev**

## References


- Kerckhoffs, A. (1883). *La cryptographie militaire.* Journal des sciences militaires, IX, 5–83.
- Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10(5), 447–474.
- Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, 37(3), 302–320.
- Angluin, D. (1978). On the complexity of minimum inference of regular sets. *Information and Control*, 39(3), 337–350.
- Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2), 87–106.
- Pitt, L., & Warmuth, M. K. (1993). The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, 40(1), 95–142.
- Kearns, M., & Valiant, L. (1994). Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1), 67–95.
- Angluin, D., & Kharitonov, M. (1995). When won't membership queries help? *Journal of Computer and System Sciences*, 50(2), 336–355.
- Bellare, M., Ristenpart, T., Rogaway, P., & Stegers, T. (2009). Format-preserving encryption. *Selected Areas in Cryptography (SAC 2009)*, LNCS 5867, 295–312.
- Durak, F. B., & Vaudenay, S. (2017). Breaking the FF3 format-preserving encryption standard over small domains. *CRYPTO 2017*, LNCS 10402, 679–707.
- Dworkin, M. (2016). *Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption.* NIST Special Publication 800-38G.
- Rivest, R. L. (1998). *Chaffing and Winnowing: Confidentiality without Encryption.* MIT Lab for Computer Science.

- Dyer, K. P., Coull, S. E., Ristenpart, T., & Shrimpton, T. (2013). Protocol misidentification made easy with format-transforming encryption. *ACM CCS 2013*, 61–72.
- Houmansadr, A., Brubaker, C., & Shmatikov, V. (2013). The parrot is dead: observing unobservable network communications. *IEEE Symposium on Security and Privacy 2013*, 65–79.
- Ajtai, M. (1996). Generating hard instances of lattice problems. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 99–108.
- Urakhchin, A. F. (2013). *Compiler Compiler System with Syntax-Controlled Runtime and Binary Application Programming Interfaces.* US Patent 8,464,232.