Supplementary Notes

Designing a minimal hairpin protospacer

In a previous study[8], we supplied small pools of double-stranded protospacers as two complementary oligos. However, because the protospacer pools we are delivering here are much more diverse, we were concerned that the complementary oligos would segregate into different cells during the electroporation. To mitigate this, we found that we could use hairpin oligo protospacers comprised of 23 bases of duplexed DNA at the core of the protospacer[16,17] with 7 uncomplemented bases at the 5' end of the bottom strand (which includes the PAM) and 5 uncomplemented bases of the 5' top strand forming a looped linker. We arrived at this minimal hairpin format by systematically testing the requirement for each of the components of the protospacer.

For instance, a protospacer adjacent motif (PAM) on one end of the protospacer both increases the efficiency of acquisition and determines the orientation of spacer insertion[8,13-15]. We found that this requirement for a PAM did not apply to both strands of the protospacer: the 'AAG' on the top strand was dispensable for the PAM effect, whereas the 'TTC' on the bottom strand was required (**Extended Data Fig. 2a**). Because the crystal structure of Cas1-Cas2 in complex with a protospacer was solved with ~23 bases of duplexed DNA at the core of the protospacer and forked, non-complementary ends on both sides[16,17], we designed an oligo hairpin that mimicked this design (**Extended Data Fig. 2b**), joining the non-complementary ends on one side with a looped linker. This initial hairpin oligo protospacer was acquired efficiently, but was cost-prohibitive to synthesize for many experiments because of its length. We therefore sought to minimize the length by removing nucleotides from the linker loop, which we found we could do without compromising acquisition, then eliminating entirely the 5' forked end (**Extended Data Fig. 2b**). This gave us a 58-base hairpin protospacer that, once optimized for concentration, was acquired efficiently into the array upon electroporation (**Extended Data Fig. 2c**). Similar to the protospacer supplied as two complementary oligos, the vast majority of minimal hairpin protospacer was acquired in a single "forward" orientation with fewer than 0.3% of forward acquired spacers starting at a base other than the intended initial G.

Potential for electroporation "bottleneck"

We also looked at whether the electroporation itself may introduce a bottleneck by quantifying the number of colony forming units (CFUs) present in one milliliter of starting culture (the volume of cells that goes into a single electroporation), before electroporation, after pre-electroporation washes, immediately post-electroporation (with either an oligo protospacer or water alone), and after one hour of post-electroporation recovery. We found that, while there is lower viability when electroporating an oligo as compared to water alone, greater than 50 million CFUs survive the electroporation, so no significant bottleneck exists for oligo library sizes in the hundreds (**Extended Data Fig. 3**).

Internal integrity of the arrays over time

To additionally assess whether internal deletions, recombinations, or mutations are common within the array after significant rounds of division, we Sanger sequenced entire arrays from the bacteria. The native array in the BL21 strain contains 12 pre-existing spacers. After electroporation and seven days of growth, we sequenced 66 complete arrays: 54 unexpanded arrays, 11 singly expanded arrays, and one double expanded array. We found no internal deletions,

recombinations, or mutations in any of these 66 arrays. This is not unexpected, as considerable spacer stability has been found in the trailing end of the array in wild bacteria[19] and archaea[20,21], despite the fact that these ancestral, trailing-end spacers are unlikely to confer immunity to common contemporary viruses.

Systematic testing of protospacer sequence parameters

To test the effect of GC%, we designed an additional 75 protospacer oligos spanning a range of GC content in the pixel-color encoding bases – fifteen protospacers each at 0, 25, 50, 75, and 100% GC. We electroporated these oligos as a single pool, then quantified the acquisition frequency of each individual oligo (**Extended Data Fig. 4b**). We found that protospacers with higher GC content were acquired at significantly higher frequencies than those with lower GC content. When the same oligos were electroporated in sub-pools within sets of a given GC percentage (**Extended Data Fig. 4b**), the same overall trend emerged – higher GC content, particularly over 50% – resulted in higher overall acquisition frequencies of the entire set.

The single pool showed very large differences between the high and low ends of the spectrum, with 0%, and 25% essentially not acquired and 50% acquired less frequently than either 75% or 100%. However, when the same sequences were sub-pooled within groups before electroporation, the differences were less substantial. Sequences with 0% GC were still almost never acquired, but, when sub-pooled, we found no significant difference between 50, 75, and 100% GC. We proceeded to electroporate a subset of 15 of these oligos individually – three per GC group. In this case, we found no difference between any of the oligos in the 25, 50, 75, or 100% group – while the 0% GC oligos were rarely acquired (**Extended Data Fig. 4c**). Moreover, within each image, oligos with a higher GC percentage were acquired at a higher frequency. Because the GC percentage and the stability of the hairpin structure are related, there was also a slight trend toward acquisition of sequences with higher hairpin stability (**Extended Data Fig. 4d**).

We next took a similar approach with mononucleotide repeats and internal PAM sequences. For each parameter, we designed 108 protospacer oligos spread over four conditions (3, 4, 6, and 8 mononucleotide repeats; 0, 1, 3, and 5 internal PAMs). When each of these sets was electroporated as a single pool, we found no difference between the oligo acquisition frequencies as a function of the condition (**Extended Data Fig. 4e,f**). However, when sub-pooled by condition, we found a reduction in acquisition frequency at the high end (6 or 8 mononucleotide repeats; 5 internal PAMs) for each. From these experiments, we conclude that, it is most beneficial to limit the range of GC% and keep the percentage at 50% or higher, while limiting high numbers of mononucleotide repeats and internal PAMs, if possible.

Supplementary Discussion

Reconstructing frame order over time to recall the GIF

To extract order information from spacers acquired by a population of bacteria over time, we began by comparing the relative order of spacer sequences within individual pixets by analyzing all single arrays that contained two spacers from the same pixet. Each spacer can be described in terms of the number of times it is found ahead of, or behind, any other spacer within individual arrays, relative to the leader. We additionally compared each spacer with respect to

the number of times it is found ahead of, or behind, newly acquired spacers from the plasmid and genome in individual arrays.

We know that the spacers will occupy physical arrangements that depend on the order that they were electroporated, with earlier spacers found further from the leader than later spacers, from which we can define a list of "ordering rules" that must apply to spacers in the actual order they were electroporated. Spacers from earlier frames will also be found more frequently in positions further from the leader than newly acquired genome- or plasmid-derived spacers, which accumulate over the course of the experiment (and spacers from later frames in the opposite position relative to genome- and plasmid-derived spacers). This information can also be used to generate an "ordering rule." Therefore, we can analyze all possible order permutations of the spacers of a given pixet to see if the relative order among the spacers in individual arrays (across a population of arrays) in any of the permutations can satisfy all possible "ordering rules" that must be true for the actual order of spacers electroporated **(Extended Data Fig. 6a**). If a spacer permutation satisfies each of the rules, we can unambiguously assign the spacers to that order – or, in this case, image frame.

However, given the large number of spacers in this complete set, the vast majority of spacers (in fact, all but one) were unable to be unambiguously assigned to frame based solely on within-pixet comparisons (due to the rarity of single arrays that contained multiple spacers from the same pixet). Therefore, we expanded the analysis to comparisons between pixets. In this case, spacers could be analyzed for physical arrangement within individual arrays relative to other spacers that had already been assigned to a frame. This allows the arrays from many more individual cells to be considered in the ordering analysis. Like the within-pixet "ordering rules," a larger set of "ordering rules" can be defined when comparing against already assigned spacers (**Extended Data Fig. 6b**). We moved through pixets beginning with those that were able to satisfy most within-pixet "ordering rules" so that the spacers we could most confidently assign would populate the reference group first. Again, all possible permutations of a set of five pixet spacers were tested against each of the between pixet rules. In this case, the permutation that satisfied the most rules was applied, and each spacer was assigned to a frame.

The entire analysis pipeline can be summarized in the following steps:

1. Create a list of all newly acquired spacers
2. For each set of five spacers with an identical pixet, analyze the relative order of the spacers in any instance where two of them are found in a single array
3. For each of spacer in that pixet-group, analyze the relative order of the spacers in any array where that spacer is found along with a newly acquired spacer from the genome or plasmid
4. Based on steps 2+3, assign spacers to frame if order is unambiguous based on all the relative orderings from many unique arrays (see **Extended Data Fig. 6a**)
5. For pixet-groups that cannot be assigned based on steps 2-4, additionally analyze the relative order of spacers in any instances of an array that contains the spacer and another newly acquired spacer that has already been assigned to a frame
6. Based on steps 2-4 and 5, assign spacers to frame as the most parsimonious ordering based on all the analyzed relative orderings from many unique arrays (see **Extended Data Fig. 6b**).

Error Sources and Error Correction:

We employed the encoding of images as a means to test our molecular recording system with real data. Though we did some optimization of encoding schemes for the image, showing that a flexible code is preferable to a rigidly defined code, it was not our intention to build the definitive encoding scheme for images using CRISPR spacers, but rather to have real data to probe and test the limits of the system. For that reason, we kept the encoding relatively concrete for illustrative purposes. However, if the only purpose of this system were to encode images, one would likely employ some form of error-correction and compression in the code. How those two goals could be achieved in this system is worth some discussion.

Errors in our recordings can be grossly attributed to three categories: one, synthesis, sequencing, or mutation (the called spacer differs from the supplied protospacer by one or more internal nucleotide changes); two, acquisition (no spacer is called from the sequences supplied or the called spacer is from the pool of supplied protospacers, but either offset from the intended initial nucleotide or acquired in the reversed direction); three, analysis (the called spacer is a partial or complete match to an e coli genome but not the reference genome used and, thus, should have been removed prior to analysis) (**Extended Data Fig. 7**). While the missing data is irrecoverable short of deeper sequencing, the other sources of error should be avoidable with error-correcting code. The major choice here is how many nucleotides one is willing to use to implement error-correction. More nucleotides will yield a better correcting code, but the addition of those nucleotides means that the entire image must be distributed over more total protospacers. Because more reads are required to accurately reconstruct images encoded across more spacers, having more protospacers increases the probability of errors of acquisition (missing data), so a balance must be achieved.

It is possible to introduce error-correction without adding any nucleotides. For instance, taking the flexible triplet code for 21 colors used the hand[F] and the GIF, each of the three triplets that code for a color could be assigned to a different cluster (A, B, and X) (**Extended Data Fig. 8e**)[31]. Rather than selecting triplets to optimize GC%, triplets could be chosen to implement an alternating cluster code (A, B, A, B, etc.) with swappable cluster X used to avoid internal PAMs and mononucleotide repeats (**Extended Data Fig. 8f-i**). Although this scheme has no net nucleotide cost, it does reduce the overall flexibility of the sequence design. Moreover, it is not the most robust error-correction, as only a subset of single base mutations would lead to pattern disruption.

An example of a more robust error-correction scheme that does require additional nucleotides is a checksum. A subset of the nucleotides of a given spacer could be devoted to checking an aspect of the rest of the spacer – if the check is not passed, the spacer should be disregarded. For instance, one could devote two bases to represent the sum of all cytosines in the image encoding section of the spacer, two bases for the sum of all thymines, and two bases for the sum of all adenines (guanines can be inferred from the other three, and two bases, counting up to sixteen, would be sufficient to cover 21 bases of image space assuming that GC% is still being optimized for) (**Extended Data Fig. 8j-l**). Thus, with six bases devoted to the checksum, the vast majority of errors would be identified. In the case of the second image, this would increase the total number of protospacers from 100 to 123. Variants of this checksum scheme, such as counting the AC% could be implemented using fewer nucleotides, although fewer errors would be caught.

Since robust error-correction increases the total nucleotide space required to encode the same image, one would likely employ some form of compression to counter this expansion. In

terms of lossless compression, run-length encoding is one option. The code would specify a pixel value and then the number of adjacent pixels that are of the same value, rather than uniquely specifying each pixel value. Since adjacent pixels in our images are often the same value, this would achieve compression. However, the fact that each individual protospacer only encodes a small number of pixels reduces the effectiveness of run-length encoding within a protospacer. The greatest benefit would be run-length encoding spread across protospacers. Unfortunately, this strategy would be highly sensitive to missing data, which could potentially disrupt large sections of the image upon recall.

A more apt option for lossless compression would be the use of a dictionary algorithm (e.g. LZW[32]/Huffman[33]/Deflation) in which the most common pixel values are encoded using the fewest bits (nucleotides) which forces rare values to be encoded using more bits. An accompanying dictionary (also supplied via protospacers) would provide the lookup table to reconstruct the pixel values. The weak spot here would be loss of the dictionary, but this could be easily circumvented by providing redundancy in the form of multiple orthogonal dictionary protospacers. If some loss of fidelity is acceptable, a lossy compression method could be applied, such as transform coding. In a sense, we could have employed this compression on the experiments presented here. Since we arrived at our encoded images by scaling down a larger image through bilinear interpolation, we could apply the reverse scaling to our resulting data images to more closely recover a larger (in bytes) image. However, this does not suit our purpose here since a portion of the error as compared to the original image would come from the lossy compression, and not directly from our experimental manipulations.

Obstacles to fully single-cell recording

To reconstruct the ordering of information over time in the GIF, we leveraged both population and single-cell information. Order determination began as a single-cell, pair-wise comparison of the order of two newly acquired spacers within a single array. We then leveraged many of these pair-wise comparisons across a population of bacteria to reconstruct the entire ordering of all 520 spacers. Given the acquisition efficiencies in this system, no individual cell would be capable of encoding all the information as we supplied it in pooled oligo protospacers. The obstacles preventing wholly single-cell reconstruction currently include the delivery of protospacers (electroporation efficiencies, which are not 100%) and acquisition efficiencies once the oligos are delivered. If one were to overcome each of these existing limitations, the next challenge would be the limitation of single genomic arrays – both because multiple acquisition events per electroporation may overwhelm a single array, and because arrays would expand to a length that could be unstable and might present significant challenges to sequencing. This problem can be practically surmounted by following the example of native systems, where single cells routinely have multiple active arrays functioning in parallel.

Information storage in DNA versus silicon

In vitro DNA is a viable alternative to information storage in silicon when a premium is placed on information density (at 10^3 times the density of flash memory) or total information lifetime (projected to be 100-2,000,000 years, depending on storage conditions, compared with ~10 years for flash memory)[25]. In vivo information storage in DNA is less dense than in vitro storage due to the physical volume of the cell as well as the presence of any non-information-encoding elements of the genome. However, like the in vitro DNA, in vivo DNA is ideal for extending the information lifetime. In vivo DNA also has the advantage that it can be repaired by

native machinery in the functioning cell, as opposed to in vitro DNA, although it is still subject to a slow increase in mutations over time and potential negative selection if the information carries a cost to the cell. In terms of replicating the information stored in DNA, in vivo DNA is cheaper to replicate than in vitro DNA (around 10-100 times less cost per copy) while also being less error prone (>100 times fewer errors per base) (see calculations below). However, the replication of in vivo DNA is around 20-30 times slower. That said, if the sole purpose is to encode information that is already in hand into DNA for long-term storage, the challenge of delivering data into cells might make in vivo information storage less attractive than in vitro. The best use case for the in vivo information recordings is one that is not yet feasible with current technology – a scenario in which cells gather and record biological information that is unknown to us before being captured by the cells. However, new approaches to biological recordings[26-28] and information encoding in nucleotides[29] are emerging rapidly, any of which could change feasibility of information storage in living cells.

Calculations of in vitro versus in vivo DNA storage parameters:

To calculate cost per copy of information for in vitro DNA, we started with the list price of KAPA HiFi master mix per 25µl reaction ($1.26). We assumed that the primer cost in a 25µl reaction is negligible. Thirty cycles of PCR under perfect conditions will generate 2^{30} copies of the input information. This is clearly an overestimate because it assumes that the PCR is perfectly efficient (which it is likely not) and remains in the exponential phase of amplification throughout all 30 cycles (which it likely does not). Nonetheless, that works out to ~1.2×10^{-7} cents per copy. To compare like-with-like, if we were to design an amplification scheme for the protospacers carrying the GIF that we encoded into cells, these copies would cost ~8.3×10^{-12} cents per bit copied. However, up to a limit, this cost per bit copied would decrease if the information in the starting material were increased. According to manufacturer provided materials, the error rate is estimated at 2.8×10^{-7} per base. Although themocycling times will vary by amplicon length, 30 minutes is the estimate that we used.

For in vivo DNA, we used the list price of pre-made liquid Luria Broth at 500ml, at 5.5 cents per ml, although assembly from components in larger volumes would clearly be much less expensive. We assumed thirty cycles of replication to, again, generate 2^{30} copies of the input information. Similar to the PCR, this may slightly overestimate the true number of copies if cells exit the exponential growth phase prior to reaching 2^{30} rounds of replication. In this case, that calculates to 5.1×10^{-9} cents per copy. Taking the GIF as an example, this would work out to ~2.5×10^{-13} cents per bit copied. However, at least up to some theoretical limit vastly beyond what we have encoded here (based on the capacity of a genome), the cost per copy is not dependent on the size of information copied, so this information would drop as more information is stored. We arrived at 2.7×10^{-9} errors per base based on an estimate of 8.9×10^{-11} errors per base per generation[34]. With a doubling time of 20 minutes, this process could take minimally 10 hours. Both the in vitro and in vivo calculations disregard the complexity of the starting material (number of individual strands or distribution among cells), which may significantly affect these estimates.

The physical size of information stored in cells is greater than a similar amount of information stored in vitro as a function of the volume of the cell. Again using the GIF as an example, the ~3 kilobytes of information can be recalled using 10^6 reads, so could be stored in ~10^6 cells, which would require 1µl of saturated culture (10^9 cell/ml), or ~2×10^4 bits per cubic millimeter. This is considerably less dense than previous in vitro information encoding

(~$5.5x10^{15}$ bits per cubic millimeter)[1]. However, as with the cost per bit copied, because the size limit is set by the volume of the cell, the bits per cubic millimeter would increase as the amount of information stored in a given genome is increased (storage density of an e coli genome itself is ~10^{16} bits per cubic centimeter[25]).

Calculations of Information Content

Images were reconstructed based on two parameters contained in the nucleotide sequences of the spacers: pixel values and the physical arrangement of those pixel values specified by the pixets. For the purposes of these calculations, we will take a conservative approach and consider the pixet information to be part of the decoding apparatus rather than stored data, thereby considering only the pixel values as information. For the $hand^R$ image, there were 56x56 (3,136) pixels, each with 4 possible values for $\log_2(4)(3,136)$=6,272 bits (784 bytes) in pixel values. The $hand^F$ image, with 30x30 (900) pixels, each with 21 possible values contained $\log_2(21)(900)$=~3,953.1 bits (~494.1 bytes) in pixel values. Finally, the GIF recording encoded 36x26 pixels over 5 frames for a total of 4,680 pixels. Each pixel had 21 possible values, for $\log_2(21)(4,680)$=~20,556 bits (~2.6 kilobytes). As in the exclusion of the pixet from the information content, this calculation of the information encoded in the GIF is a conservative estimation excluding the information about frame order contained in the relative arrangement of the spacer in the array.

Supplementary References

1 Church, G. M., Gao, Y. & Kosuri, S. Next-generation digital information storage in DNA. *Science* **337**, 1628, doi:10.1126/science.1226355 (2012).

8 Shipman, S. L., Nivala, J., Macklis, J. D. & Church, G. M. Molecular recordings by directed CRISPR spacer acquisition. *Science*, doi:10.1126/science.aaf1175 (2016).

13 Paez-Espino, D. *et al.* Strong bias in the bacterial CRISPR elements that confer immunity to phage. *Nature communications* **4**, 1430, doi:10.1038/ncomms2440 (2013).

14 Westra, E. R. *et al.* Type I-E CRISPR-cas systems discriminate target from non-target DNA through base pairing-independent PAM recognition. *PLoS genetics* **9**, e1003742, doi:10.1371/journal.pgen.1003742 (2013).

15 Shmakov, S. *et al.* Pervasive generation of oppositely oriented spacers during CRISPR adaptation. *Nucleic acids research* **42**, 5907-5916, doi:10.1093/nar/gku226 (2014).

16 Nunez, J. K., Harrington, L. B., Kranzusch, P. J., Engelman, A. N. & Doudna, J. A. Foreign DNA capture during CRISPR-Cas adaptive immunity. *Nature* **527**, 535-538, doi:10.1038/nature15760 (2015).

17 Wang, J. *et al.* Structural and Mechanistic Basis of PAM-Dependent Spacer Acquisition in CRISPR-Cas Systems. *Cell* **163**, 840-853, doi:10.1016/j.cell.2015.10.008 (2015).

19 Diez-Villasenor, C., Almendros, C., Garcia-Martinez, J. & Mojica, F. J. Diversity of CRISPR loci in Escherichia coli. *Microbiology (Reading, England)* **156**, 1351-1361, doi:10.1099/mic.0.036046-0 (2010).

20 Weinberger, A. D. *et al.* Persisting viral sequences shape microbial CRISPR-based immunity. *PLoS Comput Biol* **8**, e1002475, doi:10.1371/journal.pcbi.1002475 (2012).

21 Held, N. L., Herrera, A., Cadillo-Quiroz, H. & Whitaker, R. J. CRISPR associated diversity within a population of Sulfolobus islandicus. *PLoS One* **5**, doi:10.1371/journal.pone.0012988 (2010).

25 Zhirnov, V., Zadegan, R. M., Sandhu, G. S., Church, G. M. & Hughes, W. L. Nucleic acid memory. *Nature materials* **15**, 366-370, doi:10.1038/nmat4594 (2016).

26 Hsiao, V., Hori, Y., Rothemund, P. W. & Murray, R. M. A population-based temporal logic gate for timing and recording chemical events. *Molecular systems biology* **12**, 869, doi:10.15252/msb.20156663 (2016).

27 McKenna, A. *et al.* Whole-organism lineage tracing by combinatorial and cumulative genome editing. *Science* **353**, aaf7907, doi:10.1126/science.aaf7907 (2016).

28 Frieda, K. L. *et al.* Synthetic recording and in situ readout of lineage information in single cells. *Nature* **541**, 107-111, doi:10.1038/nature20777 (2017).

29 Erlich, Y. & Zielinski, D. DNA Fountain enables a robust and efficient storage architecture. *Science* **355**, 950-954, doi:10.1126/science.aaj2038 (2017).
31 Blawat, M. *et al.* Forward Error Correction for DNA Data Storage. *Procedia Computer Science* **80**, 1011-1022 (2016).
32 Welch, T. A. A technique for high-performance data compression. *Computer* **17**, 8-19 (1984).
33 Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE* **40**, 1098-1101 (1952).
34 Wielgoss, S. *et al.* Mutation Rate Inferred From Synonymous Substitutions in a Long-Term Evolution Experiment With Escherichia coli. *G3 (Bethesda, Md.)* **1**, 183-186, doi:10.1534/g3.111.000406 (2011).