High-speed 3D DNA-PAINT and unsupervised clustering for unlocking 3D DNA origami cryptography

G. Bimananda M. Wisna [a,d], **Daria Sukhareva**[b,d], **Jonathan Zhao**[c,d], **Deeksha Satyabola** [b,d], **Michael Matthies** [d], **Subhajit Roy**[a,d], **Petr Šulc** [b,d], **Hao Yan**[b,d,1], **and Rizal F. Hariadi** [a,d,1]

[a]Department of Physics, Arizona State University, Tempe, Arizona, USA.; [b]School of Molecular Sciences, Arizona State University, Tempe, Arizona, USA.; [c]Department of Computer Science, Arizona State University, Tempe, Arizona, USA.; [d]Center for Molecular Design and Biomimetics at the Biodesign Institute, Arizona State University, Tempe, Arizona, USA.

[1]To whom correspondence should be addressed. E-mail: hao.yan@asu.edu and rhariadi@asu.edu

This manuscript was compiled on August 29, 2023

Abstract

DNA origami cryptography, which employs nanoscale steganography to conceal information within folded DNA origami nanostructures, shows promise as a secure molecular cryptography technique due to the large 700-bit key size generated through scaffold routing and sliding and the interlacing of staple strands.[1] However, achieving the promised security, high information density, fast pattern detection, and accurate information readout requires even more secure cryptography and fast readout. Here, we advance the DNA origami cryptography protocol by demonstrating its ability to encrypt specific information in both 2D and 3D DNA origami structures, thus increasing the number of possible scaffold routings and improving the encryption key size. To this end, we used all-DNA-based steganography, enabled by high-speed 2D and 3D DNA-PAINT super-resolution imaging, which does not require protein binding to reveal the pattern, allowing for higher information density. We combined 2D and 3D DNA-PAINT data with unsupervised clustering, achieving up to 89% accuracy and high ratios of correct-to-wrong readout despite significant flexibility in the 3D DNA origami structure shown by oxDNA simulation. Furthermore, we propose design criteria that ensure complete information retrieval for the DNA origami cryptography protocol. We anticipate that this technique will be highly secure and versatile, making it an ideal solution for secure data transmission and storage via DNA.

Introduction

The information age began in the mid-20th century when transistors made of semiconductors were invented and became the building blocks of electronic devices that could perform computation and store information.[2] The realization of long-distance optical communication was inevitable due to the establishment of fiber optics and the optical amplifier.[2] As a result, a secure communication protocol implementable via semiconductor logic devices or computers and fiber optics systems became necessary. Many interdisciplinary efforts have been made to develop secure protocols, leading to the emergence of modern cryptography.[3,4] One example of modern cryptography is the AES (Advanced Encryption System) protocol, which relies on symmetric keys and is based on a substitution-permutation network with a maximum key size of 256 bits.[5] This protocol relies on the power of computation through computers to generate and maintain keys, which mainly use semiconductor materials. However, the resources of semiconductor materials, particularly silicon, Are limited[6] and requires significant electrical power to maintain the working device, specifically for data storage in many data centers.[7] DNA polymers are an attractive alternative due to their stability, programmability, high data capacity and low maintenance cost.[8–11] Several pioneering works on DNA computing[12–18] and data storage[19–22] have shown DNA to be a promising material for these applications. The key implication is that DNA molecular cryptography protocols that are applicable to DNA-based storage are crucial for secure information transmission.

Molecular cryptography protocols for two-way communications have recently been demonstrated by several groups through various chemical approaches such as harnessing optical and physical properties of molecules[23–26] and DNA by exploiting nucleotides and Watson-Crick base pairing of DNA.[27–31] The major principle of modern cryptography uses difficult mathematical problems to generate large possibilities of keys[3], which can be translated into the DNA system.[29] Zhang et al exploited the approach of DNA origami cryptography with symmetric keys for secure information transmission.[1] DNA origami with unique geometries can be formed by folding a long single-stranded scaffold DNA derived from M13mp18 bacteriophage and stapling it with hundreds of short single-stranded staple strands of DNA of a length typically below 100 nt.[32,33] The DNA origami cryptography approach relies on a difficult problem in predicting the correct scaffold routing and staple strands interlacing of DNA origami folding to form the correct geometry templates for pattern encryption.[1] The pattern encryption in the latter process is similar to steganography techniques where a message is hidden under a pattern on an object.[34] Zhang et al estimated that with an M13mp18 scaffold length of 7249 nt, the key space can go over 700 bits which is at least 2–3 times more than that of AES.[1]

Despite the powerful DNA origami cryptography technique, the readout method suffers from slow atomic force microscopy (AFM) to image patterns of DNA origami owing to the intrinsic problem of the tip scanning process as well as the limit of the resonant frequency of the tip.[35] There is a necessary compromise between resolution with the imaging time and the imaging field of view. Moreover, the complexity due to the need for biotin-streptavidin conjugations to reveal the pattern on the DNA origami may cause unwanted aggregation between them due to the multi binding sites for biotin on streptavidin.[36,37] One way to improve the readout speed and remove the extra steps of protein conjugation is by utilizing all DNA-based super-resolution imaging of DNA-PAINT that exploits the stochastic binding of single-stranded DNA (ssDNA) termed docking and short fluorophore-labeled ssDNA labeled with fluorophore termed imager strands to overcome the diffraction limit, allowing resolutions of up to 5 nm.[38] The technique enables fast super-resolution imaging with almost 100 um by 100 um imaging area to image thousands of DNA origami.[39] Previously, DNA origami and DNA-PAINT techniques have been applied for an alternative DNA storage where Dickinson et al. have developed an error-correction post-processing algorithm due to their high error rates during experimental origami folding and low detection efficiency in DNA-PAINT imaging.[22] Therefore, it is important to improve the overall strategies to achieve high information retrieval upon readout.

In this work, we report an advanced strategy of DNA origami cryptography in 2-dimensional and 3-dimensional (2D and 3D) DNA origami. The 3D DNA origami increases the decryption complexity, thus further concealing the encrypted patterns within the 3-dimensional structures. The patterns on the DNA origami are encrypted using high-speed DNA-PAINT dockings on the origami templates that enable high-speed DNA-PAINT readout.[39] We are able to maintain a high detection efficiency of docking strands that bear the pattern information on the origami using DNA-PAINT of around 90% by extending the binding length to the scaffold. Within approximately 24 min, thousands of DNA origami in one field of view, where each origami has a specific pattern with resolution of 10 nm. We combined the high-speed DNA-PAINT readout with unsupervised K-means clustering which is fast and reasonably accurate in assigning a centroid to a 2D or 3D cluster, and template alignment based on least squared distance minimization with pattern matching to extract the information. We also studied the effect of bit redundancy in the information retrieval in a high-density docking origami template. We demonstrate that 2D and 3D DNA origami encryption and decryption along with 2D and 3D DNA-PAINT imaging successfully retrieve the information with global accuracies of 70-89% despite the flexibility in our 3D DNA origami structure shown by oxDNA simulation. Furthermore, the 3D fitting of the 3D DNA-PAINT data shows better RMSD with the mean structure after oxDNA simulation as compared to the unrelaxed structure. Our method shows that DNA origami cryptography, with the advantage of difficult folding prediction, can be streamlined with fast DNA-PAINT and unsupervised clustering to achieve secure information transmission with high readout accuracy.

Results

The DNA origami cryptography protocols. The two-way DNA origami cryptography protocol using symmetric keys between a sender and receiver is depicted in Fig. 1A. As the sender, Alice needs to securely send text information "NSF" to Bob as the receiver. Using the same principles of symmetric cryptography, Alice will encrypt the message then generate keys and cipher-text. In our protocol we call the cipher text as cipher-mixtures which consist of DNA strands. The encryption consists of three main steps which are: DNA origami-templated pattern encryption, docking sequence assignment, and DNA origami encryption. The cipher-mixtures that consist of M13mp18 scaffolds and pattern-corresponding docking strands for each letter can be communicated through unsecured public communication channel whereas the keys have to be exchanged through a secure channel between Alice and Bob. The text message can be decrypted when Bob has both the keys and the cipher-mixtures where he reverses the encryption process by applying the keys to the cipher-mixtures and retrieves back the text message. The decryption follows three steps: DNA origami folding, DNA-PAINT imaging to reveal the pattern, and clustering combined with template alignment to extract the letters. Adversaries attempting to decrypt the cipher-mixtures will not be able to retrieve the information since the keys are not available.

The DNA origami encryption. The encryption of the message employs DNA origami as the encryption itself as well as a template for pattern encryption by uniquely designing the M13mp18 scaffold routing and staple strands for DNA origami geometries such as a 2D Rothemund Rectangular Origami (RRO) or a 3D wireframe cuboctahedron origami (Fig. 2B), the full staple strands are listed in Supplementary Table 1 and 2.

Fig. 1C shows the full encryption protocol whose process flows to the right (red arrows) and produces three keys. The first step is to convert the text message (first box in Fig. 1C) into binary codes (see Supplementary Table 3) that denote each letter and its position within the text (second box in Fig. 1C). The next step is to select a geometry shape of DNA origami and use the shape as the template to bear the pattern by choosing a rule of patterns that can be accommodated by the template. In our first demonstration, the 2D RRO with the size of around 90 x 60 nm is selected as the template and the pattern encryption rule is devised (third box in Fig. 1C) so that the pattern of binary codes that consist of "0" bits and "1" bits can be arranged on the RRO template (fourth box in Fig. 1C). The patterns on the DNA origami can have a resolution of sub-100 nm (in this case there are 14 to 20 nm of separation between two dots in a pattern).

The alignment markers needed to break the in-plane rotational symmetry of the RRO are shown as red circles (third box in Fig. 1C). The second step of encryption is to design a unique docking strand sequence for high-speed DNA-PAINT imaging where we adopted the sequence used by Strauss et al[39] (Fig. 1C top-middle panel); this step is called docking assignment. The third and final step is to generate the staple strands that will fold the scaffold M13mp18 into an RRO and assigning docking sequences to extend specific staple strands on the RRO DNA origami that are designated for the "1" bits in our binary codes; throughout the article, these are interchangeably called "information strands" or "docking strands". The strands on an RRO corresponding to the "0s" will have no docking sequence extension; we call these "staple strands key". The possibility space for the generation of the set of staple strands and scaffold routing are estimated to be around 700 bits for M13mp18, which will provide security for the information.[1] Finally, the cipher-mixture is generated. In our protocol, we separate the unextended staple strands by docking sequence and the information strands. In this demonstration, we have three cipher-mixtures since we have 3 letters of "N" at position 0, "S" at position 1 and "F" at position 2. Each cipher-mixture consists of universal M13mp18 scaffold strands with all corresponding information strands for each letter and position (Fig. 1A and 1C, denoted by cipher-mixture) with specific concentration (see Supplementary Table 4 for

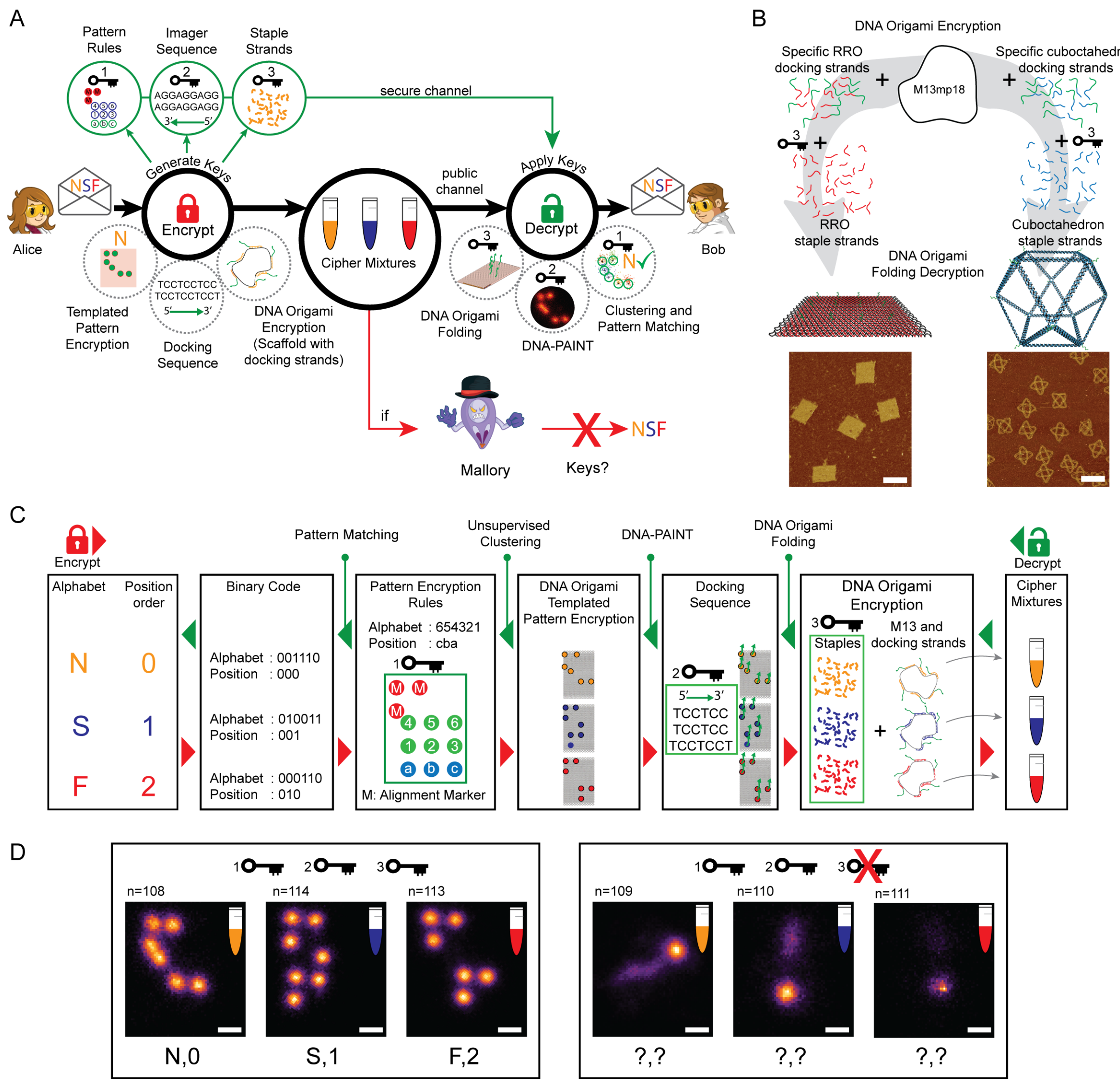


Fig. 1 **The DNA origami cryptography protocols.** (**A**) The schematic of the DNA origami cryptography protocols showing the three steps of encryption, i.e. templated pattern encryption, docking sequence, and DNA origami encryption, done by Alice to securely transmit the "NSF" message via generating three keys and a cipher mixture, as well as the three steps of decryption, i.e. DNA origami folding, DNA-PAINT, and clustering and pattern matching, done by Bob to retrieve the "NSF" message by applying the corresponding keys. The message cannot be retrieved by Mallory without the missing keys. (**B**) The schematic of DNA origami encryption where the M13mp18 scaffold can be folded into different shapes through different staple strands. A specific shape can also have different scaffold routing. AFM images are shown for two templates of 2D RRO and 3D cuboctahedron DNA origami. Scale bars are 100 nm. (**C**) The detailed protocols of the encryption and decryption processes where encryption flows to the right as denoted by the red arrows and decryption flows to the left as denoted by the green arrows. (**D**) The summed DNA-PAINT images of n patterns for each cipher mixture with three correct keys (left panel) and only two correct keys (right panel). Scale bars are 20 nm.

the mixing concentration).

The whole encryption process generates three keys which are the pattern encryption rules, the docking sequence, and the set of staple strands. The pattern encryption and docking encryption will add more layers of security. The key size for the pattern encryption in our first demonstration for 9 pattern spots is 22 bits[1]; the more the spots, the bigger the key size. For the docking encryption with 8 nt length of docking sequence with possible 4 nucleotides for each position is 16 bits. However, these extra layers of security are not the primary security because the total key sizes for these two steps are far smaller than the DNA origami encryption key size.

The DNA origami decryption and readout. The decryption is performed by applying three keys to the cipher-mixtures as shown in Fig. 1C where the process flows to the left (green arrows). The first key that needs to be applied to the cipher-mixtures is the set of DNA staple strands and with appropriate annealing (see method section for annealing details), the RRO DNA origami is formed (Fig. 1C most two right boxes). The next step is to do DNA-PAINT super-resolution imaging by applying a reverse complementary sequence of docking to be used as the imager strands that are labeled with a fluorophore, which is Cy3B. A super-resolution image of thousands of origami in one field of view can be obtained within 12 minutes to obtain 15,000 frames that are enabled by the high-speed DNA-PAINT docking sequence. We show averaged images of hundreds of DNA-PAINT of each letter in Fig. 1D left panel where we can clearly see the exact pattern as to what we designed and encrypted (compare with Fig. 1C in the fourth box from left). If we do not apply the set of staple strands key, we will obtain random patterns from DNA-PAINT and thus the true pattern is concealed (Fig. 1D right panel).

Unlike AFM, the advantage of DNA-PAINT super-resolution imaging lies in the localization-based data, which is very compatible with many clustering methods, such as supervised machine learning[40,41], Bayesian approach[42], and unsupervised machine learning for clustering.[43] In our implementation, we utilize K-means clustering to obtain the centroids of each cluster in a pattern and use the centroids for template alignment. The encoded binary can then be extracted by comparing the alignment with the pattern encryption rules to retrieve the information of "NSF". These processes show the importance of three keys in the decryption process. Without the set of staple strands key, the true pattern will not be revealed (Fig. 1D right panel). In the absence of the docking sequence knowledge, DNA-PAINT can not possibly be done to reveal the pattern and attempting to use AFM imaging modality, capturing the pattern formed by only 19 nt extension of ssDNA docking sequence will not be practically feasible as shown in Fig. 1B, bottom panels, where 12 docking extensions are not visible via AFM in the case of RRO and cuboctahedron (see Supplementary Fig S1 for more AFM images). The key of encryption pattern rules is important to translate the revealed pattern to the binary codes to retrieve the text message.

Information strands detection incorporation efficiency on 2D RRO DNA origami via DNA-PAINT. High information retention and retrieval is very important for secure information transmission. Our encryption-decryption protocol relies on the DNA origami media as well as the DNA-PAINT combined with unsupervised clustering readout. Previous work on DNA origami combined with DNA-PAINT readout for storage applications showed that this had a high error rate, thus requiring a mechanism and algorithm for robust error correction for each origami data droplet.[22] Therefore, it is crucial to have knowledge on the incorporation and detection efficiency of the information strands on the RRO origami template to be able to achieve higher information retention and retrieval. We investigate the detection efficiency of information strands on the 2D RRO DNA origami template where one information strand will be translated into one docking spot. We vary the number of spots per origami between 12, 24 and 48 as shown in Fig. 2A (full cadnano designs shown in Supplementary Fig. S2). The increasing number of spots per origami also means the smallest separation between spots decreases, the smallest separation between spots in 12, 24, and 48 spots per origami being 20 nm, 14 nm, and 10 nm respectively.

We start with an RRO with 12 spots per origami with the information strand binding section of the scaffold, which we refer to as the binder, being 32nt (Fig. 1A farthest left schematic). For the detection efficiency estimation, we follow the method developed by Strauss et al[44], and briefly present the details in Supplementary Fig. S3. We also increase the information strand binder to be 64 nt as shown in Fig. 2A (second from left schematic to farthest right schematics) to improve incorporation efficiency. It has been shown that merging the staple strands with the neighboring strands to increase the length of the initial

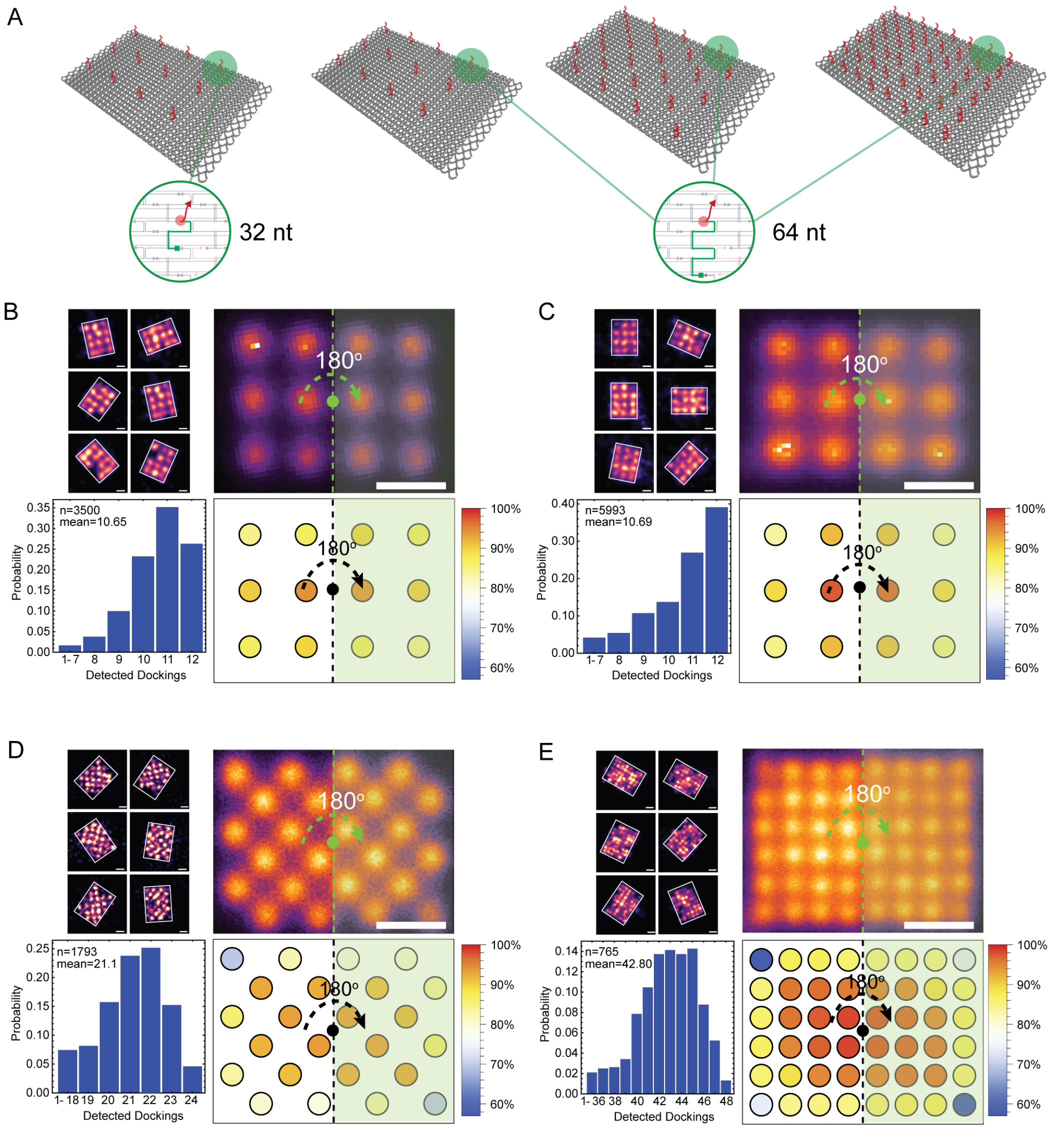


Fig. 2 **The detection efficiency of information strands on 2D RRO.** (**A**) The schematic of 2D RRO with red colored information strands: with 12 information strands with 32 nt binder (farthest left), with 12 information strands with 64 nt binder (middle left), with 24 information strands with 64-nt binder (middle right) and with 48 information strands with 64 binder (farthest right). (**B-E**) The results of 2D RRO with 12 information strands with 32 nt binder, 12 information strands with 64 nt binder, 24 information strands with 64 nt binder and 48 information strands with 64 nt binder. In each panel, examples of six individual DNA-PAINT origami with scale bars of 20 nm (left top), summed DNA-PAINT images of n origami (right top), distribution of detected docking from n origami (bottom left) and the detection efficiency of each docking position. Note that due to rotational symmetry, there will be a pair of symmetric docking positions that have the same incorporation efficiency from averaging the two efficiency values.

staple increases the melting temperature of the staple and scaffold strand complex.[45] We hypothesize that this strategy will increase the attachment lifetime of the information strands on the origami even after a series of mechanical disturbances due to DNA origami PEG purification, freeze-thaw cycles, pipetting, and mixing for sample preparation, as well as local joule heating due to high laser power density used in DNA-PAINT imaging, thus increasing the incorporation as well as the detection efficiency of the strands.

For the case of 12 spots per origami with 32nt binder, we presented our experimental results in Fig. 2B. For the case of the 64-nt binder, Figs. 2C-E show the results of origami with 12, 24, and 48 spots per origami, respectively. In each of the panels in Fig. 2B-E, we show some samples of DNA-PAINT from individual origamis in the top-left panels, the averaged image from many origamis on the top-right panels with the n number of origamis being averaged shown on each of the panels, the distribution of the number of detected docking in the bottom-left panels and the detection efficiency of each docking positions in all n number of analyzed origami in bottom-right panel. We observed that the probability of origami with full 12 dockings increased more than 50% from ∼0.25 to 0.4 when we increased the binder length from 32nt to 64-nt despite a negligible increase in the mean between the two groups from 10.65 to 10.69 (out of 12) which corresponds to detection efficiencies of 88.75% and 89.10%, respectively (Fig. 2B and 2C, bottom-left panels). Using the offset of ∼7% established by Strauss et al[44] to translate the detection efficiency to the incorporation efficiency, we obtain mean incorporation efficiencies of 95.75% and 96.10%, for the 32 and 64 nt binder respectively. This result demonstrates that the longer the binder of the information strands, the higher the chance of information strands remaining attached to the origami. The center areas of the origami for both cases show relatively high detection efficiency of >94% while the dockings at the corners and edges are prone to lower efficiency with the lowest-efficiency being the dockings at the corners (Fig. 2B and 2C, bottom-right panels). Strauss et al. also found that the docking strands that are located closer to the edges and corners of the origami suffer from lower detection efficiency as compared to the ones located around the center area of the origami, which agrees with our results.[44]

Following the improved results that we obtain with 12 docking spots per origami, we increase the density of docking spots with the 64-nt binder to investigate the effect of higher density of docking per origami on the detection efficiency (Fig. 2A, the third and fourth schematics from the left). Surprisingly, we could not get the full docking detection as the highest probability in the distribution in both cases of 24 and 48 docking spots per origami (Fig. 2D and 2E, bottom left panels) while the mean of detection efficiencies for both cases stay at 87.92% and 89.16% for 24 and 48 docking spots per origami, respectively. The translated incorporation efficiencies are 94.92% and 96.16% for a 64 nt binder of 24 and 48 docking spots per origami, respectively. These results indicate that the detection of the docking strands on the origami via DNA-PAINT (detection efficiencies) is not solely determined by the incorporation of the docking strands to the origami but also by the combined effects of the docking density and the DNA-PAINT imaging. As we increase the docking density, the DNA-PAINT imaging becomes more challenging as we have to resolve more docking with closer separation, which is 20 nm in the 12 docking spots, 14 nm in the 24 docking spots and 10 nm in the 48 docking spots (See Supplementary Fig. S2). In fact, we need to decrease the imager strand concentration from 5 nm to 1nm and increase the acquisition frame from 15,000 to 90,000 when we compare the imaging parameters of 12 docking spots and 48 docking spots, respectively (see Supplementary Table 5 for the imaging parameters of results shown in Fig. 2). The adjustments of the parameters are necessary so that we can resolve the 10 nm resolution by preventing the simultaneous blinking in higher docking density with lower imager concentration and ensuring we have enough localizations for each docking by increasing the acquisition frames.[38,46] Nevertheless, the time required to image 12 and 48 docking spots by using high-speed DNA PAINT that allows us to use 50 ms exposure time are reasonably fast at within 12 minutes and 75 minutes, respectively. Comparing the mentioned DNA-PAINT imaging time with AFM imaging time to observe highly packed and dense clusters in hundreds to thousands of origami per single field of view, the DNA-PAINT technique is 4 to 15 times faster for the readout of a pattern on origami. AFM may take 2-3 hours or more since it requires high zoom and multiple fields of view in order to achieve higher resolution.

Looking at the bottom right panels of Fig. 2D and 2E, the conclusion remains the same that the center areas of the origami have reasonably high detection via DNA-PAINT with values of >90% while the edge and corner dockings suffer from lower values. The higher the docking density, the more we have dockings with high detection values to be utilized for encrypting binary information to ensure high information retention and retrieval for our DNA origami encryption protocol. We acknowledge that our strategy to have longer binding to the scaffold for the information/docking strand limits the density of the docking number per origami to 48 which corresponds to 10 nm resolution. However, with 10 nm resolution, we can already achieve a theoretical design where we can encrypt 2^{28} ≈268.4K combinations of numbers, letters and punctuation marks forming texts and paragraphs if we can have near 100% incorporation and detection of information/docking strands (see Supplementary Fig. S4 for further details).

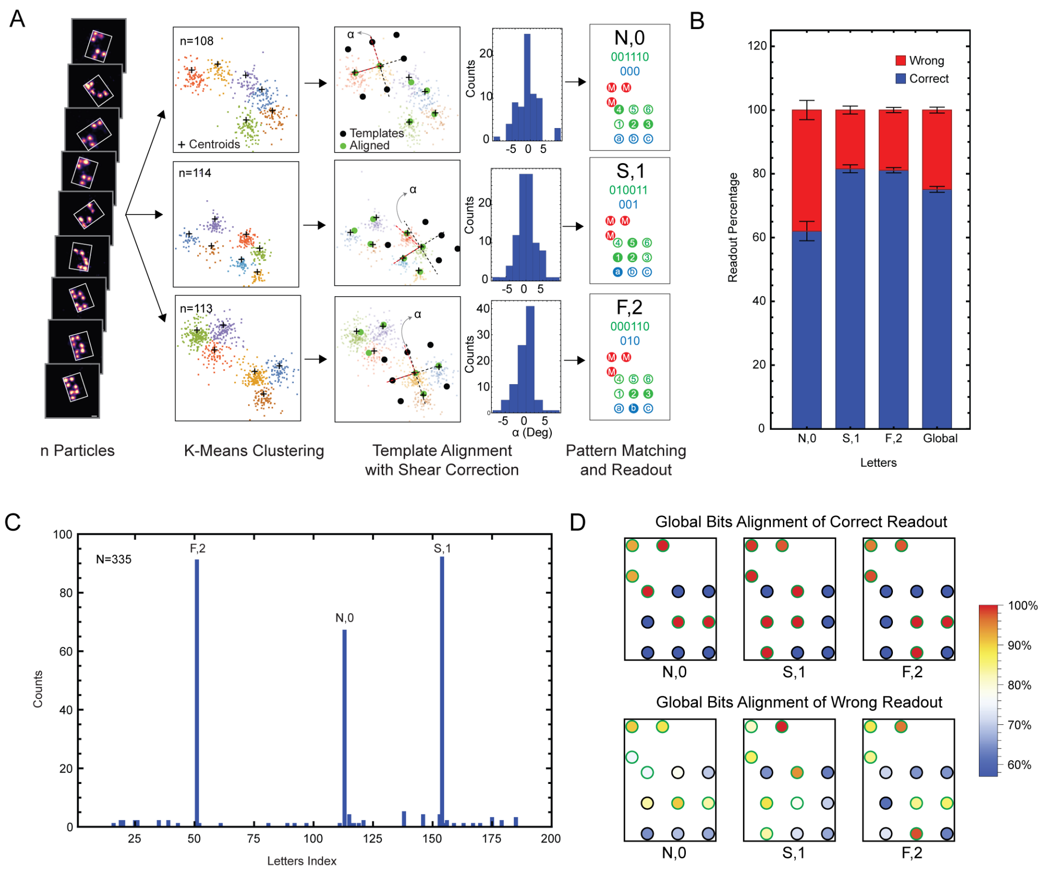


Fig. 3 **Unsupervised K-means clustering and template alignment for "NSF" data readout.** (**A**) The schematic of the process which starts with n particles (most left) followed by K-means clustering (middle left), template alignment with shear correction where the distribution of the shear angle α for each correctly read letter is shown (middle right) and pattern matching and readout (most right). (**B**) The correct and wrong readout percentage for each letter. The error bar is the standard deviation of three different runs of process shown in panel A with the same data set. (**C**) The readout result of "NSF" data presented as letter index vs counts which shows three correct-readout major peaks with small peaks of wrong readouts. (**D**) Global bits alignment of correct readout (top) and wrong readout (bottom) which shows the incorporation efficiency for each bit position using the method described in (A).

The DNA-PAINT and unsupervised clustering readout. In the first demonstration, we encrypt "NSF" text. Although we only utilize 3 patterns for "NSF", the pattern encryption rules in this demonstration can accommodate eight combinations of letters, numbers, and punctuation marks to create a text. Here we use 32 nt binder for the docking/information strands. DNA-PAINT imaging is carried out with a total time of ∼12 minutes which corresponds to 15,000 frames where we obtain several hundreds to a thousand of super-resolution images of each pattern (see supplementary Table 5 for the imaging parameters used in Fig. 3). We picked around 100 particles or origami of each pattern from the DNA-PAINT images through visual inspection for the purpose of readout accuracy analysis as shown in Fig. 3A, left panel. See supplementary Fig. S5 for all picks that are analyzed here. We employ K-means clustering on the localization-based data of each individual particle to obtain a centroid corresponding to each docking as shown in Fig. 3A, first middle panel. The K-means clustering offers a straightforward method to determine the clusters and the centroids with only one input parameter of K, which is the number of clusters to be assigned. The optimum K number for each origami can be objectively determined by using thresholding of the inertia which is commonly referred to as the elbow-method (see Method Section of K-Means Clustering and optimal Number of Clusters Selection for further details). After obtaining the centroids, we employ template alignment to align the centroids to the template matching with the design of the pattern encryption rules. Our alignment method can accommodate the alignment of the origami with some shear angle to achieve the best minimum of modified mean squared-distance cost function (see Method Section of Template Alignment, Parameter Selection and Differential Evolution and Rough for further details). The example of origami alignments of each letter along with distribution of the shear angles for correctly aligned origami are presented in Fig. 3A, right panel. Although the majority of the origami show a shear angle close to zero, some correctly aligned origami suffer from relatively large shear due to mechanical processes such as purification, mixing and pipetting during sample preparation. The bit extraction and readout are done following the template alignment process.

The analysis of individual letter accuracies along with the error bar denoting standard deviation from three different runs of the procedures of K-means clustering, template alignment by cost function minimization and bit extraction are shown in Fig. 3B. The error is attributed to the nondeterministic process of K-means as well as the cost function minimization due to many local minima that can be present. Each letter has reasonable accuracy $>60\%$ with a very small error of $\leq 3\%$ in each letter indicating the method is very robust despite the non-deterministic nature of the minimization. The global accuracy of "NSF" text shows $\sim 75\%$ with only 0.9% error, demonstrating that the DNA origami encryption and readout protocols for three letters is well-suited for secure communication between two parties. The combined readout results of the three patterns are presented in Fig. 3C where we can clearly see three main peaks of "N,0" letter, "S,1" letter and "F,2" letter. Wrong readouts occupy many small peaks. The count ratio between the lowest main peak and the highest false readout peaks is 13.4 whereas the ratio between the highest main peak and the highest false readout peak is 18.4. We further analyze the global bit alignment for correct and incorrect readouts where the correct readout shows 100% detection efficiency for the non-alignment marker dockings (Fig. 3D top panel) while the incorrect readout shows some undetected information-bearing dockings (Fig. 3D bottom panel). They also show that some origami are not correctly aligned, which is reflected from the non-zero detection in the location where no information dockings are supposed to be present.

Bit redundancy on 2D DNA origami. After successfully demonstrating 3-letter DNA origami encryption and readout with 20 nm separated docking spots, we design encryption rules that can accommodate higher information density by utilizing 48 docking spots per origami. We use 64-nt binder for the information/docking strands. We determine our docking spot assignment based on the detection efficiency results shown in Fig. 2E bottom right panel. We utilize the three spots from all four corners for orientation markers with an argument of having three redundancies to compromise the lower detection and incorporation efficiency of corner dockings as compared to the other dockings located on different areas on the origami (Fig. 4A left panels, red colored circles). We allocate 6 bits for letters, numbers and punctuation marks encryption using binary numbers. The other 6 bits are assigned for the position/order that can accommodate up to $2^6=64$ characters to form a text or a sentence. A total of 12 bits are then assigned to 12 docking spots on the origami with one additional redundancy for each bit which are boxed in red in Fig. 4A left panel. In total this scheme requires a total of 24 docking spots as shown in Fig. 4A left panels, where the green colored circles are for character bits and blue colored circles are for position bits.

We encrypt the "ASU" text which is "A,1", "S,2", and "U,3" to investigate the effect of the redundancy on readout accuracy (Fig. 4A left panels). We pick 200 particles or origami of each pattern through visual inspection for the purpose of readout accuracy analysis. We show the summed image in Fig. 4A, right panels. See supplementary Fig. S6 for all picks that are analyzed here. The total DNA-PAINT imaging time is ∼24 minutes which corresponds to 30,000 frames where we obtain several hundreds to a thousand of super-resolution images of each pattern (see supplementary Table 5 for the imaging parameters used in Fig. 4). We follow the same procedure for K-means clustering and template alignment with some alignment examples for each letter pattern shown in Fig. 4B left panel, as well as the shear angle distribution in Fig. 4B right panel. We compare the overall combined readout from three letter patterns into a single plot of the patterns by doing the readout with the bit redundancy (Fig. 4C, top panel) and without the redundancy (Fig. 4C bottom panel). Three main peaks of the three letters with their correct position appear very clear with many small peaks of wrong readout in the presence of the bit redundancy (Fig. 4C, top panel). The ratio between the lowest main peak and the highest false readout peak is 13.2 whereas the ratio between the highest main peak and the highest false readout peak is 19.3. Meanwhile, in the case of analysis without redundancy, the combined readout does not clearly show three main peaks, thus failing to retrieve the encrypted information. We can see comparable false readout peaks of letters "@,1" and "Q,2" which interfere with the final readout (Fig. 4C bottom panel).

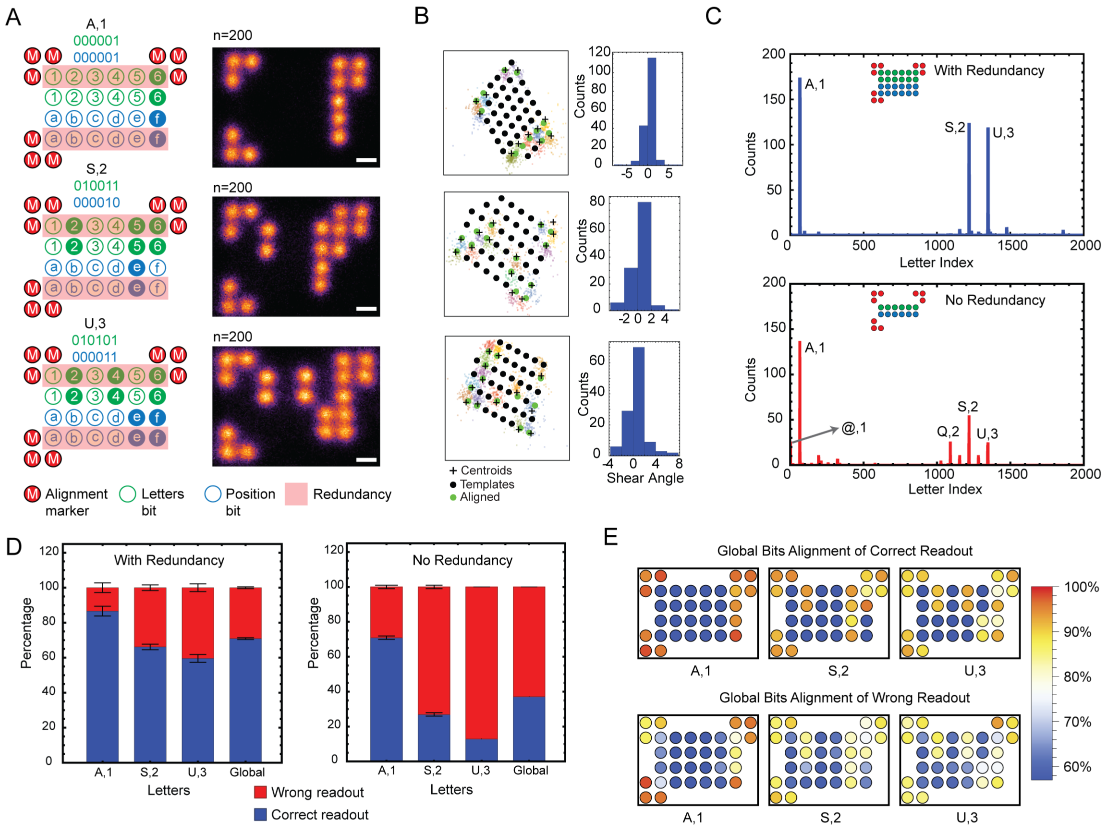


Fig. 4 **Bit redundancy on higher density 2D RRO.** (**A**)The pattern encryption rules for "ASU" dataset that shows the alignment marker, letters bit, position bit and the redundancy (left) and the summed DNA-PAINT images of three letters of "ASU" with the scale bars of 10 nm (right). (**B**) The examples of K-means clustering and template alignment for each letter (left) and the distribution of the shear angle for each letter (right). (**C**)The readout of "ASU" dataset presented as letter index vs counts which are analyzed by including the redundancy (top) and not including the redundancy (bottom).(**D**)The readout percentage of correct and wrong readout for each letter and global. The error bar is the standard deviation from three different processing runs with the same dataset. (**E**) Global bits alignment of correct readout (top) and wrong readout (bottom) which shows the incorporation efficiency for each bit position.

We analyze the accuracy for each letter in the case with and without the bit redundancy as presented in Fig. 4D left and right panels, respectively. We can clearly see that there is significant improvement of accuracy by adding one redundancy which is reflected from the global accuracy that improves from

$\sim$36% to $\sim$73%. The small standard deviation of <3% from three different runs of the K-means clustering, template alignment and bit extraction show that our method for the readout is still robust for higher docking density encryption. We also demonstrate two redundancy by utilizing all 48 docking spots and find that the two redundancy results are not significantly better than the one redundancy (see supplementary Fig. S7 and S8). Adding more redundancy does not significantly improve the results which indicate that the maximum global readout accuracy for these three letters of "ASU" that can be achieved by redundancy is around $\sim$70-75%. The false readout percentage can come from several factors such as imperfection in the imaging, which causes the closest two spots to not be well resolved, thus K-means fails to create separate centroids for these dockings. The template alignment fails in the case of significant shear and failure to resolve dockings. We also observe that the more bits are being occupied for a specific letter, the lower the accuracy which can be seen by comparing "A,1" that uses 2 bits, "S,2" that uses 4 bits and "U,3" that uses 5 bits. As the number of bits being used by a pattern increases, the accuracy decreases as shown in Fig. 4D (see supplementary Fig. S9 for further explanation).

The global bit alignment maps for each letter for correct and incorrect readout are shown in Fig. 4E. We can see that with one redundancy, the main bits do not have to be 100% present in order to retrieve back the correct letter information because of the role of the redundant bits that support the main bits. In the incorrect readout map, we can see some dockings without information strands have been falsely assigned a "1" bit.

The 3D wireframe DNA origami encryption, decryption and readout. The ability to create 3D nanostructures using DNA offers a unique avenue to expand DNA origami cryptography to increase the security due to the ability to further hide the encrypted pattern in the 3D structure. 3D DNA origami of different shapes have been realized by many groups.[33,47–50] In this demonstration, we explore the 3D shape of the cuboctahedron as the template for DNA origami encryption. Zhang et al. introduced the wireframe 3D cuboctahedron with a one-scaffolded design using M13mp18 where the structure has a dimension of around 70 nm in height.[47] The true shape and the true encrypted pattern cannot be completely revealed if the imaging modality cannot provide a 3D super-resolution imaging to resolve sub-100 nm resolution. As an example, in Fig. 1B right panel, the 3D wireframe cuboctahedron DNA origami is confused as a 2D wireframe structure by using AFM that only has 2D imaging capability, causing confusion on the true pattern encrypted on the 3D shapes (see supplementary Fig. S10 for some schematics on the confused encrypted patterns). DNA-PAINT has been able to image 3D patterns on 3D DNA origami with $\leq$110 nm z-resolution[51–53] thus it is suited to be a readout technique of 3D DNA origami encryption.

We encrypt 0407 (4th of July) on our 3D wireframe cuboctahedron DNA origami template. The docking/information strand binding length to the scaffold ranges from 52-54 nt (see supplementary Table 2). The pattern encryption is shown in Fig. 5A where we allocate 4 bits for number encryption, 3 bits for position encryption and 5 bits for alignment markers (3 filled markers and 2 empty markers) to break the in-plane (xy) rotational symmetry. The bits are all located at the vertices of the cuboctahedron in this demonstration. More bits can be placed on the edges of the structure as well. The z distance between two stacking bits is around 65-70 nm based on the ideal design by counting the number of turns of the duplex DNAs that form the edges (see supplementary Fig. S11). We imaged 43,510 frames using 50 ms exposure time which corresponds to a total acquisition time of $\sim$35 mins. We picked around 150-210 origamis for each pattern by using visual inspection (see supplementary Fig. S12)

We follow the same procedure of readout by doing K-means clustering and template alignment for each origami pick as shown in Fig. 5B. We acknowledge that it is challenging to resolve the z-separation of 60-70 nm with our setup where we cannot reliably distinguish two clusters in the localization data in an example shown Fig. 5B left panel. This can be traced back to the limit where a docking spot occupies a 3D space with z value in the range of around 50-60 nm (Fig. 5B middle panel). In addition, AFM reveals the 3D wireframe origami is actually not very rigid, further compromising the z resolution. Previously, no groups have tried 3D DNA-PAINT on the single-scaffolded 3D wireframe DNA origami thus making it hard to compare our results. Moreover, the z separation of <70 nm is beyond the state-of-the-art of the

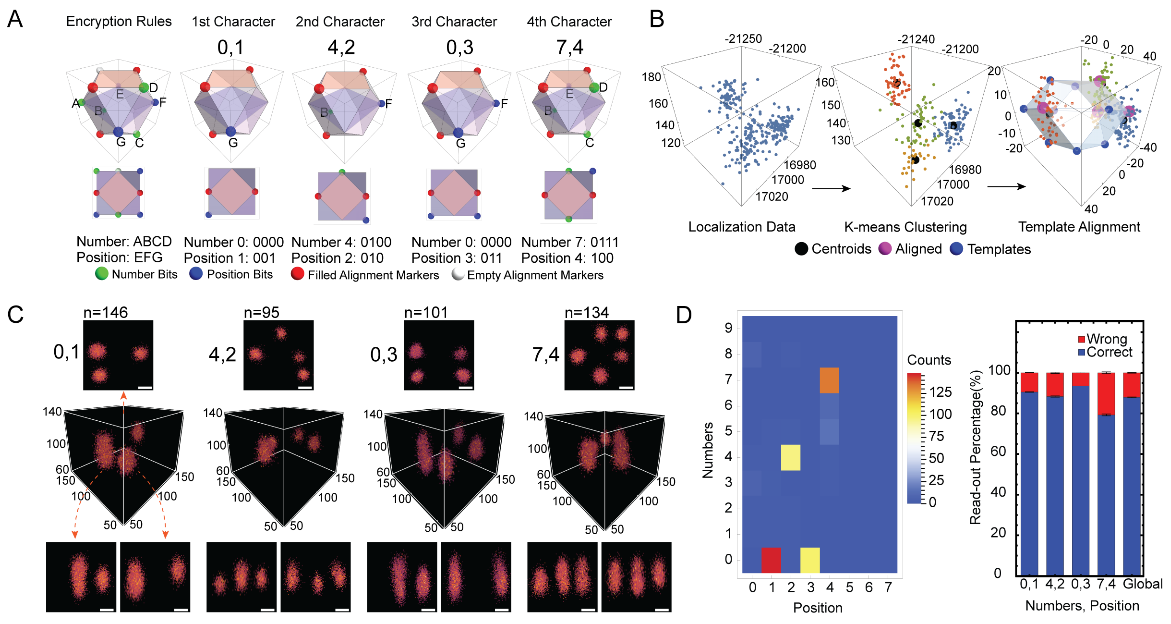
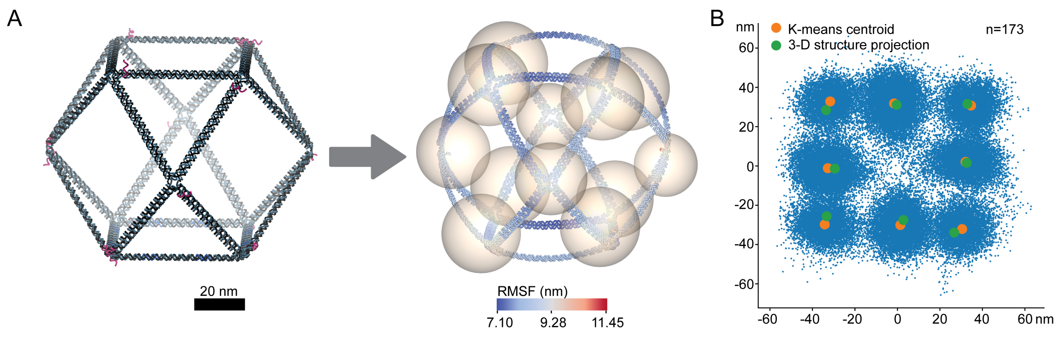


Fig. 5 **3D wireframe cuboctahedron DNA origami encryption.** (**A**)The pattern encryption rules (most left) with four encryption patterns for 4 letters shown. There are three z planes colored by transparent red (top plane), blue(middle plane) and white (bottom plane) which are occupied by the corner points. (**B**) An example of 3D DNA-PAINT localization data from a pattern on the 3D wireframe cuboctahedron DNA origami (left), an example of K-means clustering result (middle), and an example of 3D alignment with a cuboctahedron template (right). (**C**)A superposition image of correctly aligned and correctly read 4 patterns from n patterns, scale bars are 20 nm.(**D**)The result from one run of readout presented as positions vs numbers vs counts which shows 4 main peaks of letters (left) and each individual letter readout mean percentage of correct and wrong readout along with the standard deviation from 3 different runs of the same dataset of process shown in (B)

resolution of 3D DNA-PAINT that has been achieved by some groups.[51–53] However, the K-means clustering and elbow method can still cluster the localization data and assign centroids as shown in Fig. 5B middle panel with imperfect 3D DNA-PAINT localization data. We observe that the z value of the localization of two stacking clusters is distributed in the range of ∼60 nm which means the two centroids distance from the K-means clustering has a separation of only ∼30 nm which does not conform with the designed docking position (Fig. 5B middle panel). We attribute this discrepancy to the large spread of the z value of localization data for two stacking clusters as compared to the x and y spread of two neighboring clusters.

We show the summed 3D images of localization data after doing the alignment in Fig. 5C where we can see the top and side views of each pattern. The top views match the exact designs shown in Fig. 5A. We can also see a clear height difference between a cluster and 2 stacking clusters from the side views. Despite the challenge in obtaining clearly resolved two stacking dockings, we are able to align the centroids with the template (Fig. 5B). The combined readout clearly shows 4 peaks of “0,1”, “4,2”, “0,3” and “7,4” numbers with small false readout peaks in the background (Fig. 5D left panel). We employ the method in Fig. 5B with an addition of alignment marker filtering to extract the bits and retrieve the encrypted information with a global accuracy of around 89% (Fig. 5D right panel). We use alignment marker filtering because we do not have redundancy of the alignment marker as compared to the 2D origami pattern encryption rules. Missing a filled alignment marker will cause the alignment to be incorrect thus causing false readout. Our result demonstrates for the first time the 3D DNA-PAINT on a 3D wireframe cuboctahedron DNA-origami that is successfully employed as a method for 3D DNA origami encryption readout that can achieve complete information retrieval.

To further understand how flexible the cuboctahedron DNA origami structure is, we perform oxDNA



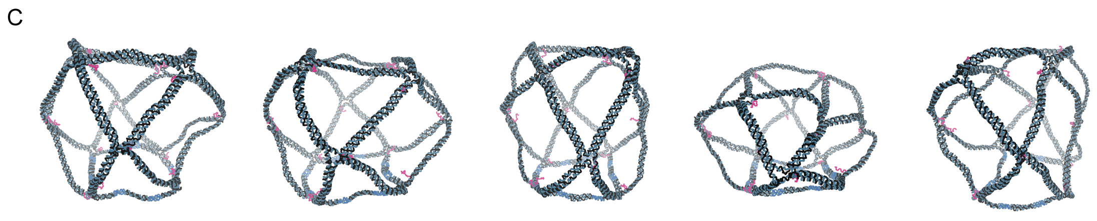


Fig. 6 **3D wireframe cuboctahedron oxDNA and DNA-PAINT alignment with the mean structure.** (**A**) The orthographic view of original unrelaxed 3D cuboctahedron structure where docking handles are colored with red (left panel). The orthographic view of mean structure after oxDNA simulation with nucleotides coloring based on the RMSF. The spheres represent the DNA-PAINT data with centers of the spheres are based on the centroids of the clusters after 3D K-means clustering and the radii are based on the average distance of each localizations to their centroids (right panel). The spheres fitting with the mean structure is obtained after doing scaling to the z value of DNA-PAINT data by 3.27. (**B**) The 2D scatter data of the DNA-PAINT localizations from n=173 particles of 3D cuboctahedron. The K-means centroids (averaged if there are two stacking centroids in z-axis) are shown by orange dots and the 2D projections of the center of the docking ssDNA (averaged if there are two stacking projections in z-axis) are shown by green dots. (**C**) Snapshots from oxDNA trajectory movie of cuboctahedron showing several deformations on the structure.

simulation on the original structure as shown in Fig. 6A left panel. The result of molecular dynamic simulation using oxDNA shows that the cuboctahedron structure is very flexible with the mean structure having root mean squared fluctuation (RMSF) ranging from 7.1 nm to 11.45 nm (Fig. 6A right panel), several snapshots from oxDNA trajectory are shown in Fig. 6C. The vertices of the structure become more rounded and the edges bend. The flexibility of the structure shown by the simulation result explains the compromise in the x,y and z axis resolution that we obtain in our 3D DNA-PAINT experimental data. We try to align our 3D DNA-PAINT localization data from 173 particles of all vertices of cuboctahedron with the docking center points in the mean structure by utilizing 3D K-means clustering with additional scaling in z axis to the 3D DNA-PAINT data. From the 3D K-means clustering, we can obtain the centroids and average radius for each clusters, see Supplementary Fig. S13A for the 3D clustering results. The final alignment for the centroids and the mean structure is shown in Fig. 6A right panel where the spheres represent each clusters with the centroid is positioned at the center of the spheres. The alignment with root mean squared distance (RMSD) of 100.88 nm requires the z-scaling of 3D DNA-PAINT data of 3.27. However, the alignment on the 2D projection of the 3D DNA-PAINT data and the mean structure produces a better results with RMSD of 26.07 nm as we can see that the green and orange dots are relatively close to each other (Fig. 6B) as compared to the one with the unrelaxed structure (see Supplementary Fig. S13D). This shows that the oxDNA simulation agrees very well with the experimental DNA-PAINT data in the xy plane without the need of scaling in xy plane. We acknowledge that the discrepancy in z-axis can be due to a problem in our 3D calibration as well as an issue with the optical alignment with our 3D set up. We also compare the 3D DNA-PAINT alignment on the mean structure with the alignment on the unrelaxed structure (see Supplementary Fig. S13B and C). The the alignment on the mean structure shows smaller 3D RMSD as compared to the alignment on the unrelaxed structure (Fig. S13B, C and E), which demonstrates the structure is indeed flexible and it agrees better with the mean structure obtained from oxDNA as compared to the unrelaxed structure.

Discussion

The results presented in this work show cryptography strategies for storing data and secure communication that are compatible with DNA systems, especially DNA origami. The approach of using DNA origami for storing data and employing DNA-PAINT super-resolution imaging techniques to reveal the pattern has been demonstrated by Dickinson et al.[22] In their digital nucleic acid memory system, they develop a robust encoding algorithm and fixed docking assignment for their data bits with only 2D rectangular origami and develop an algorithm to perform error correction.[22] We formulate high density secure data storage for secure communication in a manner that allows a large degree of freedom to choose any DNA origami geometries and to devise rules for pattern encryption for specific geometries with improved detection and fast readout. The senders can choose any variant of scaffold strands, staple strands, and unique scaffold routings, and docking sequences to achieve optimal security.

Using the results obtained in our study, design criteria which has never been discussed before in the DNA origami cryptography proof of concept work by Zhang et al[1] can be laid out to achieve high information retention and retrieval with high speed pattern readout utilizing the DNA-PAINT technique combined with unsupervised clustering and template alignment. We recommend having a docking strand binder to be at least 32 nt, with the longer the better (Fig. 1B and 1C), to be able to retrieve information through DNA-PAINT and the readout protocols. The assignment for dockings is also important. We strongly suggest utilizing the dockings in the center area of a 2D structure for higher incorporation and detection efficiency thus giving higher information retention and retrieval (Fig. 1B, 1C, 1D and 1E, bottom right panels). For the 3D origami, we encourage the user to have a z dimension of at least 100 nm due to the resolution limit of 3D DNA-PAINT. The redundancy of bits is very important when dealing with high-density docking origami (Fig. 4). We advise having an error correction code to maximize the chance of complete message retrieval. In our case of a one-redundancy code with 48 docking per origami, by using 12 dockings at corners for alignment markers, 36 dockings are available for the information bits which will then be split into 6 bits for letters plus 6 bits for the redundancy and 12 bits for positions plus 12 bits for position bits redundancy. The 12 bits correspond to a total of 4096 letters which is equivalent to 2 pages of text information (assuming 1 page can have 2000 characters).

The possibility of 2-and 3D geometries will increase the complexity of the scaffold routing and the possibility of staple strands interlacing thus we expect to have key sizes bigger than the estimated 700 bits for standard M13mp18 scaffold from Zhang et al.[1] Our protocol also improves the information density, design criteria, imaging and readout speed, and especially geometric complexity for the benefit of security. Since the key size is dependent on the length of the scaffold, another possible avenue to increase the key size is the use of several orthogonal scaffolds thus enabling bigger DNA origami. Multi-orthogonal scaffolding of bigger DNA origami has been demonstrated.[54] As an implication, with bigger DNA origami and the same information density per origami, more information strands can be incorporated thus giving higher total information that can be encrypted in a specific multi-orthogonal scaffolded DNA origami.

The state-of-the-art DNA-PAINT is 5 nm resolution that can be achieved through optimal imaging conditions and perfect drift correction.[38] This resolution is better as compared to what we demonstrate here and practically can also be achieved in our imaging system. However, to have 5 nm separation between docking strands, a shorter binding length to the scaffold needs to be applied which will reduce the retention of the docking/information strands on the origami. In our perspective, this length compromise can be solved by having modified information strands or scaffolds as demonstrated by Gerling et al and Engelhardt et al[54,55] to allow covalent linkers thus reinforcing the binding strength to maintain high information retrieval. To further increase the information density, multicolor DNA-PAINT can be carried out.[38,52] For example in the case of 2D RRO template with 48 docking strands per origami, we can use two different fluorophores such as ATTO 488 and ATTO 647 for two different high speed docking sequences. The two docking sequences can be stacked together to form the docking/information strands thus conceptually we can have additional 36 docking spots (assuming 12 docking spots from the new color are also being used for alignment markers). These docking spots can be used for 18 additional position bits and 18 bits of redundancy. This addition corresponds to a total of 28 position bits which can accommodate a total of ∼268 million characters (see Supplementary Fig. S14).

We also explore other methods of classifying the data, including other methods for detecting centroids and the unsupervised classification proposed for detecting structural heterogeneity in Huijben et al.[56] However, we use K-means due to its suitability for detecting circular clusters in 2D space as well as a spherical cluster in 3D space, its simplicity, and its fast runtime. Other methods such as DBSCAN, mean-shift clustering, and Gaussian mixture models depend on many parameters that must be fine-tuned to the data, while K-means only relies on the number of clusters which can be found by using the elbow method. The method employed by Huijben et al has a runtime that increases quadratically with the number of origami due to the requirement of the pairwise registration to create the dissimilarity matrix, while our method has a linear runtime. However, we acknowledge that using their classification method, we can achieve a better accuracy of ∼90% for the "NSF" data set by compromising the speed but it fails to give correct readout of "ASU" one redundancy dataset (see Supplementary Fig. S15). We also notice supervised learning classification models that are based on MLP[40] or based on ResNet convolutional neural networks (CNN) that we try (see Supplementary Fig. S16) that requires a training dataset for all possible permutations of the pattern for each pattern encryption rule, which makes the task impractical. For example, with our "ASU" pattern encryption rules, since we have a total of 24 bits for the information bits, we will need to generate 2^{24}=∼16.7 million patterns for the training data set. Our readout method of K-means clustering and template alignment using minimization requires no training data and can also be combined with autopicking to enable a complete automatic readout process. Further automation and speed improvement are possible by employing DNA-PAINT using deep neural networks.[57]

In conclusion, we demonstrated the DNA origami cryptography protocol by exploiting the scaffold routing possibilities to generate large key space to form 2D and 3D nanostructure with the readout enabled by high speed DNA-PAINT and fast K-means clustering to encrypt information in a secure way and retrieve the complete information with high accuracy in 2D and 3D DNA origami setting. We also perform several alternative machine learning methods for readout including ResNet CNN as well as implementing spectral clustering which demonstrates the versatility of the protocol. With the increasing use of DNA in information and communication purposes, our work along with our design criteria based on our results advances the molecular and DNA cryptography field.

Funding

The research was funded by the National Science Foundation (NSF) SemiSynBio II (2027215) and NSF SemiSynBio III (2227650).

Acknowledgements

The authors thank Chao Wang, Yinan Zhang, Pengkun Xia, Md Ashikur Rahman Laskar, Nirbhik Acharya, Ranjan Sasmal, Franky Djutanta, Ritvik Warrier, and Raymond Nucuta for their insightful discussions on our work. We also extend our thanks to Phillip R. Steen, Luciano A. Masullo, Hao Liu, and Raphael Jorand for the valuable discussions on 3D calibration and 3D DNA PAINT, and to Teun A.P.M Huijben for his helpful insights on unsupervised classification. Additionally, we thank Vaughn McGill-Adami for enabling ResNet with J.Zhao.

Correspondence

All correspondence and requests for materials should be addressed to R.F. Hariadi.

References

1. Zhang Y, Wang F, Chao J, Xie M, Liu H, Pan M, et al. DNA origami cryptography for secure communication. Nature communications. 2019;10(1):5469.
2. Riordan M, Hoddeson L. Crystal fire: The birth of the information age. WW Norton & Company; 1997.
3. Katz J, Lindell Y. Introduction to modern cryptography. CRC press; 2020.
4. Goldreich O. Modern cryptography, probabilistic proofs and pseudorandomness. vol. 17. Springer Science & Business Media; 1998.
5. National Institute of Standards and Technology. Advanced encryption standard (AES). Gaithersburg, MD; 2001.
6. Voas J, Kshetri N, DeFranco JF. Scarcity and global insecurity: the semiconductor shortage. IT Professional. 2021;23(5):78–82.
7. of Energy Efficiency O, Energy R. Data Centers and Servers;. Available from: https://www.energy.gov/eere/buildings/data-centers-and-servers.
8. Matange K, Tuck JM, Keung AJ. DNA stability: a central design consideration for DNA data storage systems. Nature communications. 2021;12(1):1358.
9. Goldman N, Bertone P, Chen S, Dessimoz C, LeProust EM, Sipos B, et al. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. Nature. 2013;494(7435):77–80.
10. Extance A. How DNA could store all the world's data. Nature. 2016;537(7618).
11. Cox JP. Long-term data storage in DNA. TRENDS in Biotechnology. 2001;19(7):247–250.
12. Liu Q, Wang L, Frutos AG, Condon AE, Corn RM, Smith LM. DNA computing on surfaces. Nature. 2000;403(13):175–178.
13. Adleman LM. Computing with DNA. Scientific american. 1998;279(2):54–61.
14. Braich RS, Chelyapov N, Johnson C, Rothemund PW, Adleman L. Solution of a 20-variable 3-SAT problem on a DNA computer. Science. 2002;296(5567):499–502.
15. Păun G, Rozenberg G, Salomaa A. DNA computing: new computing paradigms. Springer; 1998.
16. Qian L, Winfree E, Bruck J. Neural network computation with DNA strand displacement cascades. nature. 2011;475(7356):368–372.
17. Qian L, Winfree E. Scaling up digital circuit computation with DNA strand displacement cascades. Science. 2011;332(6034):1196–1201.
18. Seelig G, Soloveichik D, Zhang DY, Winfree E. Enzyme-free nucleic acid logic circuits. science. 2006;314(5805):1585–1588.
19. Church GM, Gao Y, Kosuri S. Next-generation digital information storage in DNA. Science. 2012;337(6102):1628–1628.
20. Ceze L, Nivala J, Strauss K. Molecular digital data storage using DNA. Nature Reviews Genetics. 2019;20(8):456–466.
21. Organick L, Ang SD, Chen YJ, Lopez R, Yekhanin S, Makarychev K, et al. Random access in large-scale DNA data storage. Nature biotechnology. 2018;36(3):242–248.
22. Dickinson GD, Mortuza GM, Clay W, Piantanida L, Green CM, Watson C, et al. An alternative approach to nucleic acid memory. Nature communications. 2021;12(1):2371.
23. Sarkar T, Selvakumar K, Motiei L, Margulies D. Message in a molecule. Nature communications. 2016;7(1):11374.
24. Boukis AC, Reiter K, Frölich M, Hofheinz D, Meier MA. Multicomponent reactions provide key molecules for secret communication. Nature communications. 2018;9(1):1439.
25. Lustgarten O, Motiei L, Margulies D. User authorization at the molecular scale. ChemPhysChem. 2017;18(13):1678–1687.
26. Vrijsen JH, Rubens M, Junkers T. Simple and secure data encryption via molecular weight distribution fingerprints. Polymer Chemistry. 2020;11(40):6463–6470.
27. Clelland CT, Risca V, Bancroft C. Hiding messages in DNA microdots. Nature. 1999;399(6736):533–534.
28. Leier A, Richter C, Banzhaf W, Rauhe H. Cryptography with DNA binary strands. Biosystems. 2000;57(1):13–22.
29. Gehani A, LaBean T, Reif J. DNA-based Cryptography. In: Aspects of Molecular Computing. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg; 2003. p. 167–188.
30. Cui G, Qin L, Wang Y, Zhang X. An encryption scheme using DNA technology. In: 2008 3rd International Conference on Bio-Inspired Computing: Theories and Applications. IEEE; 2008. p. 37–42.
31. Heider D, Barnekow A. DNA-based watermarks using the DNA-Crypt algorithm. BMC bioinformatics. 2007;8(1):1–10.
32. Rothemund PW. Folding DNA to create nanoscale shapes and patterns. Nature. 2006;440(7082):297–302.
33. Dietz H, Douglas SM, Shih WM. Folding DNA into twisted and curved nanoscale shapes. Science.

2009;325(5941):725–730.

34. Provos N, Honeyman P. Hide and seek: An introduction to steganography. IEEE security & privacy. 2003;1(3):32–44.
35. Hansma PK, Schitter G, Fantner GE, Prater C. High-speed atomic force microscopy. Science. 2006;314(5799):601–602.
36. Gole A, Murphy CJ. Biotin- streptavidin-induced aggregation of gold nanorods: tuning rod- rod orientation. Langmuir. 2005;21(23):10756–10762.
37. Aslan K, Luhrs CC, Pérez-Luna VH. Controlled and reversible aggregation of biotinylated gold nanoparticles with streptavidin. The Journal of Physical Chemistry B. 2004;108(40):15631–15639.
38. Dai M, Jungmann R, Yin P. Optical imaging of individual biomolecules in densely packed clusters. Nature nanotechnology. 2016;11(9):798–807.
39. Strauss S, Jungmann R. Up to 100-fold speed-up and multiplexing in optimized DNA-PAINT. Nature methods. 2020;17(8):789–791.
40. Auer A, Strauss MT, Strauss S, Jungmann R. NanoTRON: a Picasso module for MLP-based classification of super-resolution data. Bioinformatics. 2020;36(11):3620–3622.
41. Williamson DJ, Burn GL, Simoncelli S, Griffié J, Peters R, Davis DM, et al. Machine learning for cluster analysis of localization microscopy data. Nature communications. 2020;11(1):1493.
42. Fazel M, Wester MJ, Schodt DJ, Cruz SR, Strauss S, Schueder F, et al. High-precision estimation of emitter positions using Bayesian grouping of localizations. Nature Communications. 2022;13(1):7152.
43. Simoncelli S, Griffié J, Williamson DJ, Bibby J, Bray C, Zamoyska R, et al. Multi-color molecular visualization of signaling proteins reveals how C-terminal Src kinase nanoclusters regulate T cell receptor activation. Cell Reports. 2020;33(12):108523.
44. Strauss MT, Schueder F, Haas D, Nickels PC, Jungmann R. Quantifying absolute addressability in DNA origami with molecular resolution. Nature communications. 2018;9(1):1600.
45. Ramakrishnan S, Schärfen L, Hunold K, Fricke S, Grundmeier G, Schlierf M, et al. Enhancing the stability of DNA origami nanostructures: staple strand redesign versus enzymatic ligation. Nanoscale. 2019;11(35):16270–16276.
46. Schnitzbauer J, Strauss MT, Schlichthaerle T, Schueder F, Jungmann R. Super-resolution microscopy with DNA-PAINT. Nature protocols. 2017;12(6):1198–1228.
47. Zhang F, Jiang S, Wu S, Li Y, Mao C, Liu Y, et al. Complex wireframe DNA origami nanostructures with multi-arm junction vertices. Nature nanotechnology. 2015;10(9):779–784.
48. Douglas SM, Dietz H, Liedl T, Högberg B, Graf F, Shih WM. Self-assembly of DNA into nanoscale three-dimensional shapes. Nature. 2009;459(7245):414–418.
49. Han D, Pal S, Nangreave J, Deng Z, Liu Y, Yan H. DNA origami with complex curvatures in three-dimensional space. Science. 2011;332(6027):342–346.
50. Veneziano R, Ratanalert S, Zhang K, Zhang F, Yan H, Chiu W, et al. Designer nanoscale DNA assemblies programmed from the top down. Science. 2016;352(6293):1534–1534.
51. Iinuma R, Ke Y, Jungmann R, Schlichthaerle T, Woehrstein JB, Yin P. Polyhedra self-assembled from DNA tripods and characterized with 3D DNA-PAINT. science. 2014;344(6179):65–69.
52. Jungmann R, Avendaño MS, Woehrstein JB, Dai M, Shih WM, Yin P. Multiplexed 3D cellular super-resolution imaging with DNA-PAINT and Exchange-PAINT. Nature methods. 2014;11(3):313–318.
53. Auer A, Schlichthaerle T, Woehrstein JB, Schueder F, Strauss MT, Grabmayr H, et al. Nanometer-scale Multiplexed Super-Resolution Imaging with an Economic 3D-DNA-PAINT Microscope. ChemPhysChem. 2018;19(22):3024–3034.
54. Engelhardt FA, Praetorius F, Wachauf CH, Brüggenthies G, Kohler F, Kick B, et al. Custom-size, functional, and durable DNA origami with design-specific scaffolds. ACS Nano. 2019;13(5):5015–5027.
55. Gerling T, Kube M, Kick B, Dietz H. Sequence-programmable covalent bonding of designed DNA assemblies. Science Advances. 2018;4(8):eaau1157.
56. Huijben TA, Heydarian H, Auer A, Schueder F, Jungmann R, Stallinga S, et al. Detecting structural heterogeneity in single-molecule localization microscopy data. Nature communications. 2021;12(1):3791.
57. Narayanasamy KK, Rahm JV, Tourani S, Heilemann M. Fast DNA-PAINT imaging using a deep neural network. Nature Communications. 2022;13(1):5047.
58. Stahl E, Martin TG, Praetorius F, Dietz H. Facile and scalable preparation of pure and dense DNA origami solutions. Angewandte Chemie. 2014;126(47):12949–12954.
59. Huang B, Wang W, Bates M, Zhuang X. Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy. Science. 2008;319(5864):810–813.
60. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental

algorithms for scientific computing in Python. Nature methods. 2020;17(3):261–272.
61. He K, Zhang X, Ren S, Sun J. Deep residual learning. Image Recognition. 2015;7.

Materials and methods

Materials Unmodified DNA staple strands and biotinylated staple strands for 2D and 3D origami were obtained from Integrated DNA Technology (IDT). M13mp18 scaffold strands were ordered from Bayou Biolabs. Imager strands were made from amine-modified DNA strands (IDT DNA) and conjugated with Cy3B in-house using NHS Ester coupling. Cy3B-NHS Ester fluorophores are ordered from General Electric Healthcare (Codes: PA63101). High-performance liquid chromatography (HPLC) is used to purify the imager strands. Chemicals are supplied by Sigma Aldrich. Glass coverslips (48466-205) and microscope slides (16004-430) are purchased from VWR. Kapton Tape for the flow chamber is purchased from Bertech (PPTDE-2 310-787-0337).

2D and 3D DNA origami folding and purification To fold the 2D and 3D DNA origami, we mixed M13mp18 scaffold strands with staple strands, biotinylated strands, and corresponding docking strands for each pattern in 1X TAE 12.5 mM Mg^{2+} buffer. The scaffold strand concentration was 20 nM, while the staple strands were in 10X concentration compared to the scaffold. To optimize the incorporation of docking and biotinylated strands, we used 50–60X concentration, as suggested by Strauss et al.[44]. The mixture was annealed using a protocol of ramping up to 80 ^oC and keeping it for 5 minutes, followed by a slow ramp down to 4 ^oC with a rate of 3 minutes 12 seconds per degree Celsius, based on the protocol by Schnitzbauer et al.[46]. After annealing, we purified the DNA origami solution using a PEG precipitation method with a PEG buffer (15% (m/v) PEG 8000, 12.5mM $MgCl_2$, 505mM NaCl, 5mM Tris-HCl, 1mM EDTA, pH=8.0) for three rounds, as described by Stahl et al.[58]. Finally, we redispersed the solution in 1XTAE 12.5mM Mg^{2+} buffer and stored it at -20^{o}C. The entire process encompasses the decryption of DNA origami through origami folding.

AFM imaging of 2D and 3D DNA origami Multimode AFM from Bruker is used to image DNA origami after PEG purification. 3 uL of 1-2 nM DNA origami solution is deposited on a freshly cleaved mica surface on an AFM metal sample slab. Then 60 uL of 1XTAE 12.5 mM MgCl and 4 uL of 0.2 M $NiCl_2$ solution are added to immobilize the origami onto the mica surface. This sample is imaged in the AFM using fluid mode with SCANASYST-FLUID+ model tip by Bruker. The image is acquired with DI-AFM Bruker and processed using NanoScope Analysis AFM software.

Pattern readout by 2D and 3D DNA-PAINT super-resolution imaging and image processing DNA-PAINT super-resolution imaging is performed generally by following the detailed protocols described in the work by Schnitzbauer et al.[46] Briefly, DNA origamis are immobilized on a BSA-biotin-streptavidin coated coverslip forming a flow chamber with the microscope slide glass through double sided kapton tape. Buffer A+ (10 mM Tris-HCl, 100 mM NaCl, 0.05% (v/v) Tween 20, pH 8.0) is used to dilute BSA-Biotin and streptavidin to 1 mg/ml and 0.5 mg/ml concentration, respectively. Buffer B+(5 mM Tris-HCl, 10 mM $MgCl_2$, 1 mM EDTA, 0.05% (v/v) Tween 20, pH 8.0) is used to dilute DNA origami to experimental concentrations. Buffer B+ is also used to dilute the imager strands (AGGAGGA/3' Cy3B/) to experimental concentrations. Oxygen scavenger solutions PCA, PCD and Trolox with final concentrations of 1.25X PCA, 1X PCD and 1X Trolox are mixed with the imager strands to make the final imaging solution. The experimental conditions for each figure are described in the supplementary Table 4.

We use Oxford Nanoimager (ONI) of Benchtop Nanoimager S Mark II with a total internal reflection fluorescence (TIRF) set up. Cy3B is excited by using a 532 nm laser with a power density ranging from 800 W/cm^2 to 1250 W/cm^2. A 549-623 nm Band pass filter is installed on the emission path to select the Cy3B emission. For the NSF dataset, an Olympus objective with 100X magnification and 1.4 NA with oil immersion is used. For the ASU and 0407 dataset, an Olympus objective with 100X magnification and

1.49 NA with immersion oil is used. A Hamamatsu ORCA-Flash4.0 V3 digital sCMOS camera is used to acquire the DNA-PAINT movies with camera exposure time of 50 ms. The Nanoimager is equipped with z-lock autofocus with piezo stage. For 3D DNA-PAINT, an additional 3D lens is inserted in the optical path that will modify the point spread function (PSF) from circular to elliptical PSF which is dependent on the z distance of the emitter from the focal spot through astigmatism.[46,51,53,59] The z-calibration is done by using 2D RRO by scanning the z from -500 nm to +500 nm with an increment of 10 nm made possible by the piezo stage. A standard 3D calibration curve (see supplementary Fig 17) is generated through the Picasso Localize module which is used as a calibration curve to process the 3D DNA-PAINT movies. The imaging conditions for each figure are described in the supplementary Table 5.

The DNA-PAINT movies are then processed using FIJI imageJ to crop the image into 256 px by 256 px. Then, each movie is fed into the Picasso Localize module to fit the PSF into precise localization data (see supplementary Table 6 for Picasso Localize module parameters). The Picasso Render module is used to perform multiple redundant cross correlation (RCC) drift corrections to obtain the final super-resolution images. Picasso Filter module is used to outlier localizations based on x,y localization precision and localization background. The whole processing is done in an Alienware Desktop Computer with Intel Core i7-6800K CPU 32 GB RAM and NVIDIA GeForce GTX 1080 graphics card.

2D DNA-PAINT Incorporation efficiency analysis The incorporation efficiency analysis follows the method described by Strauss et al.[44] Briefly, Picasso processed localization data of super-resolution images of the patterns using Picasso pick automatic function. All picks are aligned using the Picasso Average module. The aligned picks are then unfolded in the Picasso Render module to get arrays of picks that have been aligned in the same orientation. Then, it is fed into a Matlab script that analyzes the incorporation efficiency (see supplementary Fig. S3).

2-and 3D data clustering, template alignment and analysis The protocol follows the pipeline as shown in Fig. 3A. The patterns on each origami are picked by using visual inspection on the Picasso Render module. The picks are then used as an input for the following processing of K-means clustering that assigns centroids based on the optimal number cluster followed by cluster filtering. Then the centroids are aligned with the corresponding template based on the pattern encryption rules. The processing is done in a System 76 Thelio Desktop with AMD Ryzen 9 5950X 16-Core Processor, 32GB RAM and NVIDIA GeForce RTX 3070 GPU. The detailed explanations for each process follow below.

K-Means Clustering is done using the Scipy Python package[60] for the 2D data and a Matlab function for the 3D data. In the 3D data, an extra data filtering using DBSCAN function in Matlab is performed by empirically assigning the radius (ϵ) to be 7 and the minimum number of neighbor points to determine the core to be 4. In the implementation, K-Means clustering is used to cluster localizations into N through M clusters, where N and M are chosen according to the pattern encryption template. K-means clustering takes one parameter K for the number of clusters. **The optimal K value** is determined by using the elbow method by running K-Means for (M - N) times over each origami, initially with N clusters and increasing by 1 every iteration until M clusters where M is determined by the maximum docking numbers for specific templates. The inertias, centroids, and cluster memberships resulting from the K-Means clustering are recorded at every iteration. Centroids are defined as the center of each cluster found by K-Means. Inertias are defined as the squared distances between each localization and its corresponding centroid. The differences between the inertias, at each iteration, termed as the gradients, are calculated then negated and normalized to have a maximum value of 1. The optimal number of clusters is chosen by comparing these differences, and locating the point at which the difference negligibly improves. The optimal number of clusters is the point at which the gradient value starts to saturate to a value of 0.95 or more (maximum value is 1) in the case of 3D data or closest to some threshold T_I in the case of 2D data. See Supplementary Table 7 for the parameters. The elbow method is a relatively standard practice.

Cluster filtering is done after K-Means clustering over the 2D data. Outlying clusters are filtered out based on size, which is defined as the number of localizations belonging to that cluster. To do this, the mean size of all clusters is calculated, and the size threshold is calculated by multiplying the mean size by some value T_S. Any clusters with a size smaller than threshold size are discarded, and localizations belonging to those clusters are also filtered out.

Template creation is performed based on the pattern encryption rules of 2D and 3D DNA origami. Four templates are created for the grids used in the NSF dataset, the ASU one redundancy dataset, the ASU two redundancy dataset, and the 3D cuboctahedron 0407 dataset. These templates use a prior knowledge of the possible binding site locations and orientation marker positions. Each point in the template represents the approximate (x, y) position of the binding site in an idealized origami. Note all binding sites are included, not just the binding sites used in any particular pattern.

2D Template Alignment. The centroids, template, cluster sizes, template weights, and orientation markers are used to calculate a transform of the template that minimizes the following cost function modified from Euclidean squared distance formula: $C = \frac{\sqrt{D+1}}{n}\sum_{i=1}^{n}(|A_i - B_i| * P_i * S_i)^2\}$

A_i and B_i are the Euclidean coordinates of the ith centroid and the closest point in the transformed template to A_i by Euclidean distance, respectively. Si is the number of localizations in the ith centroid's cluster. $|A_i$-$B_i|$ is the Euclidean distance between A_i and B_i. P_i is the weighting of B_i defined as $P_i = W_0$ if B_i is an orientation marker, otherwise $P_i = 1$. Finally, D is the number of orientation markers that are not the closest point to any centroid.

We assume there is a higher probability that binding sites lie closer to larger clusters, hence the cost penalty is larger if the template is misaligned from larger clusters. In addition, the orientation markers are very important for readout, so these points in the template are weighted at W_0>1, and any missing orientation markers are penalized with the D+1 term. Besides these additions, the cost function is a squared nearest-neighbor distance metric.

The parameters of the transform include rotation by an angle θ, X or horizontal scaling, Y or vertical scaling, X translation, and Y translation, which are applied in that order. The rotation prior to scaling allows the transformed templates to shear in response to distorted origami. At each iteration of optimization, the current transform is applied to the template, then the cost is calculated.

C is minimized according to any optimization method, and the final transform is applied to the template. For the NSF dataset, we first align the template to the origami by translating and rotating the template at fixed intervals around the origami until the minimal distance is achieved. The rough transform is then fine-tuned using the L-BFGS-B method of the minimize function from Scipy.[60] For the ASU 1-redundancy and 2-redundancy datasets, we directly optimize the transform using the differential evolution method of the minimize function from Scipy. For both datasets, at each iteration of the Scipy algorithm, all transformation parameters are simultaneously optimized and applied in the order specified above. **Parameter selection** is done by empirically finding good values for N and M. If origami with higher numbers of binding sites are imaged, we may need a higher value of M to account for false positives. T_I and T_S were selected through grid search. W_O was empirically selected. The best method for alignment for each dataset was empirically selected. See Supplementary Table 7 for the parameters.

Finally, the closest points in the template to each of the centroids are converted into a binary string using our knowledge of the template. In the case of the repetition datasets, if any of the binding sites corresponding to a single bit are activated, then the bit is set to 1. The decimal value of this binary string can then be converted using our alphabet encoding.

3D Template Alignment. Using the same general idea in 2D template alignment. We use centroids' and the 3D cuboctahedron template's Euclidean coordinates to minimize the Euclidean squared distances of each centroid to the closest point on the template using the formula: $C = \sum_{i=1}^{n}(|A_i - B_i|)^2\}$

Initially, the 3D centroids and the 3D template are brought close to each other by aligning the center of mass of the 3D centroids and the 3D template. The template is then z-scaled to fix the z scale discrepancy that happened due to centroid assignment by K-means clustering. Then, the minimization is done through a series of steps of x and y translation then rotation about the z-axis with a total of 360 ^{o}. Z translation can also be performed but it will cause the running time to be longer. The z translation does not significantly affect our alignment result due to the small difference in the z value of the template and pattern center of mass. In all steps, the C is calculated and the minimum value is taken for the final transform. After the alignment, we filter out the origamis that are missing at least one orientation marker and analyze the results. The binary conversion is done similarly as in 2D by using the aligned patterns by using the knowledge of the binary translation that we use in pattern encryption rules.

oxDNA simulation and 3D alignment of unrelaxed and mean structure with experimental DNA-PAINT data. The simulation was carried out at 25°C in a periodic boundary cube of length around 168nm using oxDNA2 forcefield and langevin thermostat, for 1.52×10^{-5}s. An average of 6 such replicas were considered for the study, accounting for a total simulation time of 9×10^{-5}s.

The PAINT data is projected into the 2D x-y plane and fitted with simulation results to obtain the best 2D possible configuration. With the optimal 2D configuration conserves the z-axis. It is then scaled with small factor to obtain desirable structure. To obtain the best 2D configuration, the average configuration from the simulation is rotated such that the plane formed by the biotin strands is at the bottom of the frame and the normal of the plane is perpendicular to the xy plane, similar to the experimental setup. The center of mass of the 12 docking handles are projected to the x-y plane and fitted with the projection of the PAINT data points. The geometry of the origami is such that when the structure is projected onto x-y plane, the 4 point at extreme top along z axis and 4 at the very bottom will overlap making them indistinguishable to clustering algorithm thus making them to look only as 8 clusters instead of 12 clusters. The projected data points from the DNA-PAINT was clustered into 8 points instead of 12 using K-means algorithm. The two configurations are fitted using SVD superimposition technique (Fig. 6B for mean structure and Fig. S13D for unrelaxed structure). The 2D configuration generated in this step is remained unchanged through out the later procedures.

With 2D alignment being fixed, the z-axis of PAINT data is increased by small factor and clustered into 12 clusters. The sum of the RMSD between the centroids and the mean position of the docking handles is minimized for an optimal z-axis scaling factor (Fig. S13E). To further confirm that the K-means produced meaningful clusters, the 12 centroids generated are again projected into 2D plane and total RMSD between the current configuration and the old 2d configuration was noted. If the resultant RMSD exceed the uncertainty, in this case was taken to be the RMSD from SVD 2D fitting from previous step, the model is rejected. These results are visually confirmed as well to validate our findings.The K-means algorithm for the 3D fitting has an accuracy of 84%, which is predicted by executing the algorithm multiple time with random seeds and checking the projection 2D rmsd generated after 3D clustering (Fig. S13E). With the final check of 2D RMSD, the algorithm is successfully able to figure out the scaling factor within a deviation of 6%.

Code Availability The scripts and code for processing the data can be found on https://github.com/Jonathanzhao02/smlm_classification2d and https://github.com/gwisna

Supporting Information

High-speed 3D DNA-PAINT and unsupervised clustering for unlocking 3D DNA origami cryptography

G. Bimananda M. Wisna [a,d], **Daria Sukhareva**[b,d], **Jonathan Zhao**[c,d], **Deeksha Satyabola** [b,d], **Michael Matthies** [d], **Subhajit Roy**[a,d], **Petr Šulc** [b,d], **Hao Yan**[b,d,1], **and Rizal F. Hariadi** [a,d,1]

[a]Department of Physics, Arizona State University, Tempe, Arizona, USA.; [b]School of Molecular Sciences, Arizona State University, Tempe, Arizona, USA.; [c]Department of Computer Science, Arizona State University, Tempe, Arizona, USA.; [d]Center for Molecular Design and Biomimetics at the Biodesign Institute, Arizona State University, Tempe, Arizona, USA.

[1]To whom correspondence should be addressed. E-mail: hao.yan@asu.edu and rhariadi@asu.edu

Contents

S1. Materials and Methods

Materials Unmodified DNA staple strands and biotinylated staple strands for 2- and 3D origami are ordered from Integrated DNA Technology (IDT). M13mp18 scaffold strands are ordered from Bayou Biolabs. Imager strands are obtained from amine-modified DNA strands, which are ordered from IDT and conjugated with Cy3B in-house using NHS Ester coupling. Cy3B-NHS Ester fluorophores are ordered from General Electric Healthcare (Codes: PA63101). High-performance liquid chromatography (HPLC) is utilized for the purification of imager strands. Chemicals are supplied by Sigma Aldrich. Coverslip (Cat. No. 48466-205) and microscope slides (Cat. No. 16004-430) are purchased from VWR. Kapton Tape for the flow chamber is purchased from Bertech (PPTDE-2 310-787-0337).

Methods **Unsupervised classification as described by Huijben et al.**[56] We follow the protocol for unsupervised classification to classify a pre-labeled mixture of "NSF" dataset into several classes. The superparticles of each class are then fed into our template alignment method to read out the bit. We count the number of each label in each class to calculate the accuracy (see supplementary Fig. 14).

ResNet CNN supervised classification. To generate the synthetic dataset, we utilize Picasso's Simulate and Render modules to generate 75x75 images of each of the 26 letters from the alphabet. Images are then filtered using the root mean squared distance between each pixel and the center of mass of the image, totaling 41096 images. The dataset is then split into a training dataset of 22749 images and a testing dataset of 18347 images. Twenty percent of the training set is further split into a validation dataset used in selecting the final model.

Our ResNet implementation uses transfer learning on ResNet-50.[61] We replace the final layer with a linear layer that gives 12 outputs with a sigmoid activation, one for each binding site in the "NSF" pattern encryption template. A threshold of 0.5 is used to distinguish between bits that are on and off, which are then decoded into the corresponding letter.

The network is trained with the Pytorch implementation of the Adam optimizer at a learning rate of 1e-3 for the linear layer and 1e-4 for the other layers for 20 epochs. A cosine annealing learning rate scheduler and binary cross entropy loss is used. After training, we select the epoch with the lowest loss over the validation set as the final model. This model is then used to run predictions over the testing dataset (see Supplementary Fig. S15).

S2. Supplementary Tables

Name	Sequence	Note
0[47]1[31]	AGAAAGGAACAACTAAAGGAATTCAAAAAAA	Core staple
1[96]3[95]	AAACAGCTTTTTGCGGGATCGTCAACACTAAA	Core staple
2[111]0[112]	AAGGCCGCTGATACCGATAGTTGCGACGTTAG	Core staple
3[160]4[144]	TTGACAGGCCACCACCAGAGCCGCGATTTGTA	Core staple
5[160]6[144]	GCAAGGCCTCACCAGTAGCACCATGGGCTTGA	Core staple
6[239]4[240]	GAAATTATTGCCTTTAGCGTCAGACCGGAACC	Core staple
7[224]9[223]	AACGCAAAGATAGCCGAACAAACCCTGAAC	Core staple
8[239]6[240]	AAGTAAGCAGACACCACGGAATAATATTGACG	Core staple
9[224]11[223]	AAAGTCACAAAATAAACAGCCAGCGTTTTA	Core staple
10[239]8[240]	GCCAGTTAGAGGGTAATTGAGCGCTTTAAGAA	Core staple
11[224]13[223]	GCGAACCTCCAAGAACGGGTATGACAATAA	Core staple
13[96]15[95]	TAGGTAAACTATTTTTGAGAGATCAAACGTTA	Core staple
0[79]1[63]	ACAACTTTCAACAGTTTCAGCGGATGTATCGG	Core staple
1[128]4[128]	TGACAACTCGCTGAGGCTTGCATTATACCAAGCGCGATGATAAA	Core staple
2[143]1[159]	ATATTCGGAACCATCGCCCACGCAGAGAAGGA	Core staple
3[224]5[223]	TTAAAGCCAGAGCCGCCACCCTCGACAGAA	Core staple
5[224]7[223]	TCAAGTTTCATTAAAGGTGAATATAAAAGA	Core staple
6[271]4[272]	ACCGATTGTCGGCATTTTCGGTCATAATCA	Core staple
7[248]9[255]	GTTTATTTTGTCACAATCTTACCGAAGCCCTTTAATATCA	Core staple
8[271]6[272]	AATAGCTATCAATAGAAAATTCAACATTCA	Core staple
9[256]11[255]	GAGAGATAGAGCGTCTTTCCAGAGGTTTTGAA	Core staple
10[271]8[272]	ACGCTAACACCCACAAGAATTGAAAATAGC	Core staple
11[256]13[255]	GCCTTAAACCAATCAATAATCGGCACGCGCCT	Core staple
13[128]15[127]	GAGACAGCTAGCTGATAAATTAATTTTTGT	Core staple
0[111]1[95]	TAAATGAATTTTCTGTATGGGATTAATTTCTT	Core staple
1[160]2[144]	TTAGGATTGGCTGAGACTCCTCAATAACCGAT	Core staple
2[175]0[176]	TATTAAGAAGCGGGGTTTTGCTCGTAGCAT	Core staple
4[79]2[80]	GCGCAGACAAGAGGCAAAAGAATCCCTCAG	Core staple
6[47]4[48]	TACGTTAAAGTAATCTTGACAAGAACCGAACT	Core staple
7[32]9[31]	TTTAGGACAAATGCTTTAAACAATCAGGTC	Core staple
8[47]6[48]	ATCCCCCTATACCACATTCAACTAGAAAAATC	Core staple
9[32]11[31]	TTTACCCCAACATGTTTTAAATTTCCATAT	Core staple
10[47]8[48]	CTGTAGCTTGACTATTATAGTCAGTTCATTGA	Core staple
11[32]13[31]	AACAGTTTTGTACCAAAAACATTTTATTTC	Core staple
12[79]10[80]	AAATTAAGTTGACCATTAGATACTTTTGCG	Core staple
13[160]14[144]	GTAATAAGTTAGGCAGAGGCATTTATGATATT	Core staple
0[175]0[144]	TCCACAGACAGCCCTCATAGTTAGCGTAACGA	Core staple
1[192]4[192]	GCGGATAACCTATTATTCTGAAACAGACGATTGGCCTTGAAGAGCCAC	Core staple
2[207]0[208]	TTTCGGAAGTGCCGTCGAGAGGGTGAGTTTCG	Core staple
4[143]3[159]	TCATCGCCAACAAAGTACAACGGACGCCAGCA	Core staple
6[79]4[80]	TTATACCACCAAATCAACGTAACGAACGAG	Core staple
7[56]9[63]	ATGCAGATACATAACGGGAATCGTCATAAATAAAGCAAAG	Core staple
8[79]6[80]	AATACTGCCCAAAAGGAATTACGTGGCTCA	Core staple
9[64]11[63]	CGGATTGCAGAGCTTAATTGCTGAAACGAGTA	Core staple
10[79]8[80]	GATGGCTTATCAAAAAGATTAAGAGCGTCC	Core staple
11[64]13[63]	GATTTAGTCAATAAAGCCTCAGAGAACCCTCA	Core staple
12[143]11[159]	TTCTACTACGCGAGCTGAAAAGGTTACCGCGC	Core staple
13[192]15[191]	GTAAAGTAATCGCCATATTTAACAAAACTTTT	Core staple
0[239]1[223]	AGGAACCCATGTACCGTAACACTTGATATAA	Core staple
1[224]3[223]	GTATAGCAAACAGTTAATGCCCAATCCTCA	Core staple
2[239]0[240]	GCCCGTATCCGGAATAGGTGTATCAGCCCAAT	Core staple
4[207]2[208]	CCACCCTCTATTCACAAACAAATACCTGCCTA	Core staple
6[111]4[112]	ATTACCTTTGAATAAGGCTTGCCCAAATCCGC	Core staple
7[96]9[95]	TAAGAGCAAATGTTTAGACTGGATAGGAAGCC	Core staple
8[111]6[112]	AATAGTAAACACTATCATAACCCTCATTGTGA	Core staple
9[96]11[95]	CGAAAGACTTTGATAAGAGGTCATATTTCGCA	Core staple
10[111]8[112]	TTGCTCCTTTCAAATATCGCGTTTGAGGGGGT	Core staple
11[96]13[95]	AATGGTCAACAGGCAAGGCAAAGAGTAATGTG	Core staple
12[207]10[208]	GTACCGCAATTCTAAGAACGCGAGTATTATTT	Core staple
13[224]15[223]	ACAACATGCCAACGCTCAACAGTCTTCTGA	Core staple
0[271]1[255]	CCACCCTCATTTTCAGGGATAGCAACCGTACT	Core staple
1[256]4[256]	CAGGAGGTGGGGTCAGTGCCTTGAGTCTCTGAATTTACCGGGAACCAG	Core staple
2[271]0[272]	GTTTTAACTTAGTACCGCCACCCAGAGCCA	Core staple
4[271]2[272]	AAATCACCTTCCAGTAAGCGTCAGTAATAA	Core staple
6[143]5[159]	GATGGTTTGAACGAGTAGTAAATTTACCATTA	Core staple
7[120]9[127]	CGTTTACCAGACGACAAAGAAGTTTTGCCATAATTCGA	Core staple
8[143]7[159]	CTTTTGCAGATAAAAACCAAAATAAAGACTCC	Core staple
9[128]11[127]	GCTTCAATCAGGATTAGAGAGTTATTTTCA	Core staple
10[143]9[159]	CCAACAGGAGCGAACCAGACCGGAGCCTTTAC	Core staple
11[128]13[127]	TTTGGGGATAGTAGTAGCATTAAAAGGCCG	Core staple

12[271]10[272]	TGTAGAAATCAAGATTAGTTGCTCTTACCA	Core staple
13[256]15[255]	GTTTATCAATATGCGTTATACAAACCGACCGT	Core staple
1[32]3[31]	AGGCTCCAGAGGCTTTGAGGACACGGGTAA	Core staple
2[47]0[48]	ACGGCTACAAAAGGAGCCTTTAATGTGAGAAT	Core staple
3[32]5[31]	AATACGTTTGAAAGAGGACAGACTGACCTT	Core staple
5[32]7[31]	CATCAAGTAAAACGAACTAACGAGTTGAGA	Core staple
6[175]4[176]	CAGCAAAAGGAAACGTCACCAATGAGCCGC	Core staple
7[160]8[144]	TTATTACGAAGAACTGGCATGATTGCGAGAGG	Core staple
8[175]6[176]	ATACCCAACAGTATGTTAGCAAATTAGAGC	Core staple
9[160]10[144]	AGAGAGAAAAAAATGAAAATAGCAAGCAAACT	Core staple
10[175]8[176]	TTAACGTCTAACATAAAAACAGGTAACGGA	Core staple
11[160]12[144]	CCAATAGCTCATCGTAGGAATCATGGCATCAA	Core staple
13[32]15[31]	AACGCAAAATCGATGAACGGTACCGGTTGA	Core staple
14[47]12[48]	AACAAGAGGGATAAAAATTTTTAGCATAAAGC	Core staple
1[64]4[64]	TTTATCAGGACAGCATCGGAACGACACCAACCTAAAACGAGGTCAATC	Core staple
2[79]0[80]	CAGCGAAACTTGCTTTCGAGGTGTTGCTAA	Core staple
3[96]5[95]	ACACTCATCCATGTTACTTAGCCGAAAGCTGC	Core staple
5[96]7[95]	TCATTCAGATGCGATTTTAAGAACAGGCATAG	Core staple
6[207]4[208]	TCACCGACGCACCGTAATCAGTAGCAGAACCG	Core staple
7[184]9[191]	CGTAGAAAATACATACCGAGGAAACGCAATAAGAAGCGCA	Core staple
8[207]6[208]	AAGGAAACATAAAGGTGGCAACATTATCACCG	Core staple
9[192]11[191]	TTAGACGGCCAAATAAGAAACGATAGAAGGCT	Core staple
10[207]8[208]	ATCCCAATGAGAATTAACTGAACAGTTACCAG	Core staple
11[192]13[191]	TATCCGGTCTCATCGAGAACAAGCGACAAAAG	Core staple
13[64]15[63]	TATATTTTGTCATTGCCTGAGAGTGGAAGATT	Core staple
14[79]12[80]	GCTATCAGAAATGCAATGCCTGAATTAGCA	Core staple
14[111]12[112]	GAGGGTAGGATTCAAAAGGGTGAGACATCCAA	Core staple
15[96]17[95]	ATATTTTGGCTTTCATCAACATTATCCAGCCA	Core staple
16[111]14[112]	TGTAGCCATTAAAATTCGCATTAAATGCCGGA	Core staple
17[224]19[223]	CATAAATCTTTGAATACCAAGTGTTAGAAC	Core staple
19[32]21[31]	GTCGACTTCGGCCAACGCGCGGGGTTTTTC	Core staple
21[56]23[63]	AGCTGATTGCCCTTCAGAGTCCACTATTAAAGGGTGCCGT	Core staple
22[79]20[80]	TGGAACAACCGCCTGGCCCTGAGGCCCGCT	Core staple
23[96]22[112]	CCCGATTTAGAGCTTGACGGGGAAAAAGAATA	Core staple
16[271]14[272]	CTTAGATTTAAGGCGTTAAATAAAGCCTGT	Core staple
14[143]13[159]	CAACCGTTTCAAATCACCATCAATTCGAGCCA	Core staple
15[128]18[128]	TAAATCAAAATAATTCGCGTCTCGGAAACCAGGCAAAGGGAAGG	Core staple
16[143]15[159]	GCCATCAAGCTCATTTTTTAACCACAAATCCA	Core staple
18[47]16[48]	CCAGGGTTGCCAGTTTGAGGGGACCCGTGGGA	Core staple
19[96]21[95]	CTGTGTGATTGCGTTGCGCTCACTAGAGTTGC	Core staple
21[96]23[95]	AGCAAGCGTAGGGTTGAGTGTTGTAGGGAGCC	Core staple
22[111]20[112]	GCCCGAGAGTCCACGCTGGTTTGCAGCTAACT	Core staple
23[128]23[159]	AACGTGGCGAGAAAGGAAGGGAAACCAGTAA	Core staple
18[271]16[272]	CTTTTACAAAATCGTCGCTATTAGCGATAG	Core staple
14[175]12[176]	CATGTAATAGAATATAAAGTACCAAGCCGT	Core staple
15[160]16[144]	ATCGCAAGTATGTAAATGCTGATGATAGGAAC	Core staple
16[175]14[176]	TATAACTAACAAAGAACGCGAGAACGCCAA	Core staple
18[79]16[80]	GATGTGCTTCAGGAAGATCGCACAATGTGA	Core staple
19[160]20[144]	GCAATTCACATATTCCTGATTATCAAAGTGTA	Core staple
21[120]23[127]	CCCAGCAGGCGAAAAATCCCTTATAAATCAAGCCGGCG	Core staple
22[143]21[159]	TCGGCAAATCCTGTTTGATGGTGGACCCTCAA	Core staple
23[160]22[176]	TAAAAGGGACATTCTGGCCAACAAAGCATC	Core staple
20[271]18[272]	CTCGTATTAGAAATTGCGTAGATACAGTAC	Core staple
14[207]12[208]	AATTGAGAATTCTGTCCAGACGACTAAACCAA	Core staple
15[192]18[192]	TCAAATATAACCTCCGGCTTAGGTAACAATTTCATTTGAAGGCGAATT	Core staple
16[207]14[208]	ACCTTTTTATTTTAGTTAATTTCATAGGGCTT	Core staple
18[111]16[112]	TCTTCGCTGCACCGCTTCTGGTGCGGCCTTCC	Core staple
19[224]21[223]	CTACCATAGTTTGAGTAACATTTAAAATAT	Core staple
21[160]22[144]	TCAATATCGAACCTCAAATATCAATTCCGAAA	Core staple
22[175]20[176]	ACCTTGCTTGGTCAGTTGGCAAAGAGCGGA	Core staple
23[192]22[208]	ACCCTTCTGACCTGAAAGCGTAAGACGCTGAG	Core staple
22[271]20[272]	CAGAAGATTAGATAATACATTTGTCGACAA	Core staple
14[239]12[240]	AGTATAAAGTTCAGCTAATGCAGATGTCTTTC	Core staple
15[224]17[223]	CCTAAATCAAAATCATAGGTCTAAACAGTA	Core staple
16[239]14[240]	GAATTTATTTAATGGTTTGAAATATTCTTACC	Core staple
18[143]17[159]	CAACTGTTGCGCCATTCGCCATTCAAACATCA	Core staple
20[79]18[80]	TTCCAGTCGTAATCATGGTCATAAAAGGGG	Core staple
21[184]23[191]	TCAACAGTTGAAAGGAGCAAATGAAAAATCTAGAGATAGA	Core staple
22[207]20[208]	AGCCAGCAATTGAGGAAGGTTATCATCATTTT	Core staple
23[224]22[240]	GCACAGACAATATTTTTGAATGGGGTCAGTA	Core staple
14[271]12[272]	TTAGTATCACAATAGATAAGTCCACGAGCA	Core staple

15[256]18[256]	GTGATAAAAAGACGCTGAGAAGAGATAACCTTGCTTCTGTTCGGGAGA	Core staple
17[32]19[31]	TGCATCTTTCCCAGTCACGACGGCCTGCAG	Core staple
18[175]16[176]	CTGAGCAAAAATTAATTACATTTTGGGTTA	Core staple
20[143]19[159]	AAGCCTGGTACGAGCCGGAAGCATAGATGATG	Core staple
21[224]23[223]	CTTTAGGGCCTGCAACAGTGCCAATACGTG	Core staple
22[239]20[240]	TTAACACCAGCACTAACAACTAATCGTTATTA	Core staple
23[256]22[272]	CTTTAATGCGCGAACTGATAGCCCCACCAG	Core staple
15[32]17[31]	TAATCAGCGGATTGACCGTAATCGTAACCG	Core staple
16[47]14[48]	ACAAACGGAAAAGCCCCAAAAACACTGGAGCA	Core staple
17[96]19[95]	GCTTTCCGATTACGCCAGCTGGCGGCTGTTTC	Core staple
18[207]16[208]	CGCGCAGATTACCTTTTTTAATGGGAGAGACT	Core staple
20[207]18[208]	GCGGAACATCTGAATAATGGAAGGTACAAAAT	Core staple
21[248]23[255]	AGATTAGAGCCGTCAAAAAACAGAGGTGAGGCCTATTAGT	Core staple
23[32]22[48]	CAAATCAAGTTTTTTGGGGTCGAAACGTGGA	Core staple
0[143]1[127]	TCTAAAGTTTTGTCGTCTTTCCAGCCGACAA	Core staple
15[64]18[64]	GTATAAGCCAACCCGTCGGATTCTGACGACAGTATCGGCCGCAAGGCG	Core staple
16[79]14[80]	GCGAGTAAAAATATTTAAATTGTTACAAAG	Core staple
17[160]18[144]	AGAAAACAAAGAAGATGATGAAACAGGCTGCG	Core staple
18[239]16[240]	CCTGATTGCAATATATGTGAGTGATCAATAGT	Core staple
21[32]23[31]	TTTTCACTCAAAGGGCGAAAAACCATCACC	Core staple
22[47]20[48]	CTCCAACGCAGTGAGACGGGCAACCAGCTGCA	Core staple
23[64]22[80]	AAAGCACTAAATCGGAACCCTAATCCAGTT	Core staple
0[207]1[191]	TCACCAGTACAAACTACAACGCCTAGTACCAG	Core staple
4[47]2[48]	GACCAACTAATGCCACTACGAAGGGGGTAGCA	Core staple
20[47]18[48]	TTAATGAACTAGAGGATCCCCGGGGGGTAACG	Core staple
4[111]2[112]	GACCTGCTCTTTGACCCCCAGCGAGGGAG	Core staple
20[111]18[112]	CACATTAAAATTGTTATCCGCTCATGCGGGCC	Core staple
4[175]2[176]	CACCAGAAAGGTTGAGGCAGGTCATGAAAG	Core staple
20[175]18[176]	ATTATCATTCAATATAATCCTGACAATTAC	Core staple
4[239]2[240]	GCCTCCCTCAGAATGGAAAGCGCAGTAACAGT	Core staple
20[239]18[240]	ATTTTAAAATCAAAATTATTTGCACGGATTCG	Core staple
12[47]10[48]	TAAATCGGGATTCCCAATTCTGCGATATAATG	Core staple
12[111]10[112]	TAAATCATATAACCTGTTTAGCTAACCTTTAA	Core staple
12[175]10[176]	TTTTATTTAAGCAAATCAGATATTTTTTGT	Core staple
12[239]10[240]	CTTATCATTCCCGACTTGCGGGAGCCTAATT	Core staple
4[63]6[56]	TTTTATAAGGGAACCGGATATTCATTACGTCAGGACGTTGGGAA	Core staple
4[127]6[120]	TTTTTTGTGTCGTGACGAGAAACACCAAATTTCAACTTTAAT	Core staple
4[191]6[184]	TTTTCACCCTCAGAAACCATCGATAGCATTGAGCCATTTGGGAA	Core staple
4[255]6[248]	TTTTAGCCACCACTGTAGCGCGTTTTCAAGGGAGGGAAGGTAAA	Core staple
18[63]20[56]	TTTTATTAAGTTTACCGAGCTCGAATTCGGGAAACCTGTCGTGC	Core staple
18[127]20[120]	TTTTGCGATCGGCAATTCCACACAACAGGTGCCTAATGAGTG	Core staple
18[191]20[184]	TTTTATTCATTTTTGTTTGGATTATACTAAGAAACCACCAGAAG	Core staple
18[255]20[248]	TTTTAACAATAACGTAAAACAGAAATAAAAATCCTTTGCCCGAA	Core staple
biotin-4[63]6[56]	/5Biosg/TTTTATAAGGGAACCGGATATTCATTACGTCAGGACGTTGGGAA	Biotinylated staple
biotin-4[127]6[120]	/5Biosg/TTTTTTGTGTCGTGACGAGAAACACCAAATTTCAACTTTAAT	Biotinylated staple
biotin-4[191]6[184]	/5Biosg/TTTTCACCCTCAGAAACCATCGATAGCATTGAGCCATTTGGGAA	Biotinylated staple
biotin-4[255]6[248]	/5Biosg/TTTTAGCCACCACTGTAGCGCGTTTTCAAGGGAGGGAAGGTAAA	Biotinylated staple
biotin-18[63]20[56]	/5Biosg/TTTTATTAAGTTTACCGAGCTCGAATTCGGGAAACCTGTCGTGC	Biotinylated staple
biotin-18[127]20[120]	/5Biosg/TTTTGCGATCGGCAATTCCACACAACAGGTGCCTAATGAGTG	Biotinylated staple
biotin-18[191]20[184]	/5Biosg/TTTTATTCATTTTTGTTTGGATTATACTAAGAAACCACCAGAAG	Biotinylated staple
biotin-18[255]20[248]	/5Biosg/TTTTAACAATAACGTAAAACAGAAATAAAAATCCTTTGCCCGAA	Biotinylated staple
4[47]2[48]-2T-R1	GACCAACTAATGCCACTACGAAGGGGGTAGCATTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
20[47]18[48]-2T-R1	TTAATGAACTAGAGGATCCCCGGGGGGTAACGTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
4[111]2[112]-2T-R1	GACCTGCTCTTTGACCCCCAGCGAGGGAGTTATTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
20[111]18[112]-2T-R1	CACATTAAAATTGTTATCCGCTCATGCGGGCCTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
4[175]2[176]-2T-R1	CACCAGAAAGGTTGAGGCAGGTCATGAAAGTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
20[175]18[176]-2T-R1	ATTATCATTCAATATAATCCTGACAATTACTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
4[239]2[240]-2T-R1	GCCTCCCTCAGAATGGAAAGCGCAGTAACAGTTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
20[239]18[240]-2T-R1	ATTTTAAAATCAAAATTATTTGCACGGATTCGTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
12[47]10[48]-2T-R1	TAAATCGGGATTCCCAATTCTGCGATATAATGTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
12[111]10[112]-2T-R1	TAAATCATATAACCTGTTTAGCTAACCTTTAATTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
12[175]10[176]-2T-R1	TTTTATTTAAGCAAATCAGATATTTTTTGTTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
12[239]10[240]-2T-R1	CTTATCATTCCCGACTTGCGGGAGCCTAATTTTTTCCTCCTCCTCCTCCTCCT	20 nm docking strand with 32 nt binder
6[47]2[48]-2T-R1	TACGTTAAAGTAATCTTGACAAGAACCGAACTGACCAACTAATGCCACTACGAAGGGGGTAGCATTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
14[47]10[48]-2T-R1	AACAAGAGGGATAAAAATTTTTAGCATAAAGCTAAATCGGGATTCCCAATTCTGCGATATAATGTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
22[47]18[48]-2T-R1	CTCCAACGCAGTGAGACGGGCAACCAGCTGCATTAATGAACTAGAGGATCCCCGGGGGGTAACGTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
6[111]2[112]-2T-R1	ATTACCTTTGAATAAGGCTTGCCCAAATCCGCGACCTGCTCTTTGACCCCCAGCGAGGGAGTTATTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder

14[111]10[112]-2T-R1	GAGGGTAGGATTCAAAAGGGTGAGACATCCAATAAATCATATAACCTGTTTAGCTAACCTTTAATTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
22[111]18[112]-2T-R1	GCCCGAGAGTCCACGCTGGTTTGCAGCTAACTCACATTAAAATTGTTATCCGCTCATGCGGGCCTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
6[175]2[176]-2T-R1	CAGCAAAAGGAAACGTCACCAATGAGCCGCCACCAGAAAGGTTGAGGCAGGTCATGAAAGTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
14[175]10[176]-2T-R1	CATGTAATAGAATATAAAGTACCAAGCCGTTTTTATTTAAGCAAATCAGATATTTTTTGTTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
22[175]18[176]-2T-R1	ACCTTGCTTGGTCAGTTGGCAAAGAGCGGAATTATCATTCAATATAATCCTGACAATTACTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
6[239]2[240]-2T-R1	GAAATTATTGCCTTTAGCGTCAGACCGGAACCGCCTCCCTCAGAATGGAAAGCGCAGTAACAGTTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
14[239]10[240]-2T-R1	AGTATAAAGTTCAGCTAATGCAGATGTCTTTCCTTATCATTCCCGACTTGCGGGAGCCTAATTTTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
22[239]18[240]-2T-R1	TTAACACCAGCACTAACAACTAATCGTTATTAATTTTAAAATCAAAATTATTTGCACGGATTCGTTTCCTCCTCCTCCTCCTCCT	10 nm, 14 nm, and 20 nm docking strand with 64 nt binder
10[47]6[48]-2T-R1	CTGTAGCTTGACTATTATAGTCAGTTCATTGAATCCCCCTATACCACATTCAACTAGAAAAATCTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
18[79]14[80]-2T-R1	GATGTGCTTCAGGAAGATCGCACAATGTGAGCGAGTAAAAATATTTAAATTGTTACAAAGTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
21[56]22[80]-2T-R1	GCTGATTGCCCTTCAGAGTCCACTATTAAAGGGTGCCGTAAAGCACTAAATCGGAACCCTAATCCAGTTTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
6[143]6[144]-2T-R1	GATGGTTTGAACGAGTAGTAAATTTACCATTAGCAAGGCCTCACCAGTAGCACCATGGGCTTGATTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
14[143]14[144]-2T-R1	CAACCGTTTCAAATCACCATCAATTCGAGCCAGTAATAAGTTAGGCAGAGGCATTTATGATATTTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
22[143]22[144]-2T-R1	TCGGCAAATCCTGTTTGATGGTGGACCCTCAATCAATATCGAACCTCAAATATCAATTCCGAAATTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
10[207]6[208]-2T-R1	ATCCCAATGAGAATTAACTGAACAGTTACCAGAAGGAAACATAAAGGTGGCAACATTATCACCGTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
18[207]14[208]-2T-R1	CGCGCAGATTACCTTTTTTAATGGGAGAGACTACCTTTTTATTTTAGTTAATTTCATAGGGCTTTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
21[184]22[208]-2T-R1	ACAGTTGAAAGGAGCAAATGAAAAATCTAGAGATAGAACCCTTCTGACCTGAAAGCGTAAGACGCTGAGTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
10[271]6[272]-2T-R1	ACGCTAACACCCACAAGAATTGAAAATAGCAATAGCTATCAATAGAAAATTCAACATTCATTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
18[271]14[272]-2T-R1	CTTTTACAAAATCGTCGCTATTAGCGATAGCTTAGATTTAAGGCGTTAAATAAAGCCTGTTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
21[248]22[272]-2T-R1	GATTAGAGCCGTCAAAAAACAGAGGTGAGGCCTATTAGTCTTTAATGCGCGAACTGATAGCCCCACCAGTTTCCTCCTCCTCCTCCTCCT	10 nm, and 14 nm docking strand with 64 nt binder
10[111]6[112]-2T-R1	TTGCTCCTTTCAAATATCGCGTTTGAGGGGGTAATAGTAAACACTATCATAACCCTCATTGTGATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
10[143]10[144]-2T-R1	CCAACAGGAGCGAACCAGACCGGAGCCTTTACAGAGAGAAAAAAATGAAAATAGCAAGCAAACTTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
10[175]6[176]-2T-R1	TTAACGTCTAACATAAAAACAGGTAACGGAATACCCAACAGTATGTTAGCAAATTAGAGCTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
10[239]6[240]-2T-R1	GCCAGTTAGAGGGTAATTGAGCGCTTTAAGAAAAGTAAGCAGACACCACGGAATAATATTGACGTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
14[207]10[208]-2T-R1	AATTGAGAATTCTGTCCAGACGACTAAACCAAGTACCGCAATTCTAAGAACGCGAGTATTATTTTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
14[271]10[272]-2T-R1	TTAGTATCACAATAGATAAGTCCACGAGCATGTAGAAATCAAGATTAGTTGCTCTTACCATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
14[79]10[80]-2T-R1	GCTATCAGAAATGCAATGCCTGAATTAGCAAAATTAAGTTGACCATTAGATACTTTTGCGTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
18[111]14[112]-2T-R1	TCTTCGCTGCACCGCTTCTGGTGCGGCCTTCCTGTAGCCATTAAAATTCGCATTAAATGCCGGATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
18[143]18[144]-2T-R1	CAACTGTTGCGCCATTCGCCATTCAAACATCAAGAAAACAAAGAAGATGATGAAACAGGCTGCGTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
18[175]14[176]-2T-R1	CTGAGCAAAAATTAATTACATTTTGGGTTATATAACTAACAAAGAACGCGAGAACGCCAATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
18[239]14[240]-2T-R1	CCTGATTGCAATATATGTGAGTGATCAATAGTGAATTTATTTAATGGTTTGAAATATTCTTACCTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
18[47]14[48]-2T-R1	CCAGGGTTGCCAGTTTGAGGGGACCCGTGGGAACAAACGGAAAAGCCCCAAAAACACTGGAGCATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
2[143]2[144]-2T-R1	ATATTCGGAACCATCGCCCACGCAGAGAAGGATTAGGATTGGCTGAGACTCCTCAATAACCGATTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
21[184]22[208]-2T-R1	ACAGTTGAAAGGAGCAAATGAAAAATCTAGAGATAGAACCCTTCTGACCTGAAAGCGTAAGACGCTGAGTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
21[224]22[240]-2T-R1	CTTTAGGGCCTGCAACAGTGCCAATACGTGGCACAGACAATATTTTTGAATGGGGTCAGTATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
21[32]22[48]-2T-R1	TTTTCACTCAAAGGGCGAAAAACCATCACCCAAATCAAGTTTTTTGGGGTCGAAACGTGGATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder

21[96]22[112]-2T-R1	AGCAAGCGTAGGGTTGAGTGTTGTAGGGAGCCCCCGATTTAGAGCTTGACGGGGAAAA AGAATATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
22[207]18[208]-2T-R1	AGCCAGCAATTGAGGAAGGTTATCATCATTTTGCGGAACATCTGAATAATGGAAGGT ACAAAATTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
22[271]18[272]-2T-R1	CAGAAGATTAGATAATACATTTGTCGACAACTCGTATTAGAAATTGCGTAGATACAG TACTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
22[79]18[80]-2T-R1	TGGAACAACCGCCTGGCCCTGAGGCCCGCTTTCCAGTCGTAATCATGGTCATAAAA GGGGTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
23[128]22[176]-2T-R1	AACGTGGCGAGAAAGGAAGGGAAACCAGTAATAAAAGGGACATTCTGGCCAACAAA GCATCTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
6[207]2[208]-2T-R1	TCACCGACGCACCGTAATCAGTAGCAGAACCGCCACCCTCTATTCACAAACAAATA CCTGCCTATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
6[271]2[272]-2T-R1	ACCGATTGTCGGCATTTTCGGTCATAATCAAAATCACCTTCCAGTAAGCGTCAGT AATAATTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder
6[79]2[80]-2T-R1	TTATACCACCAAATCAACGTAACGAACGAGGCGCAGACAAGAGGCAAAAGAATCCCT CAGTTTCCTCCTCCTCCTCCTCCT	10 nm docking strand with 64 nt binder

Table S1 **2D RRO strands.** Scaffold is MP13mp18. Core staples are all the strands that form the structure. Biotinylated staples are modified with biotin modification for immobilization of origami on coverslip surface through BSA-Biotin-Streptavidin-Biotin-DNA origami arrangement. To fold a 20 nm pattern on 2D RRO with 64 binder, we mix scaffold strands, the 10 nm, 14 nm, and 20 nm strands with 64 nt binder along with core staples (positions corresponding to 20 nm docking positions and biotin positions should be excluded beforehand) and biotinylated staple strands. To fold a 14 nm pattern on 2D RRO with 64 binder, we mix scaffold strands, the 10 nm, 14 nm, and 20 nm strands with 64 nt binder, the 10 nm, and 14 nm docking strands with 64 nt binder along with core staples (positions corresponding to 20 nm docking positions and biotin positions should be excluded beforehand). To fold a 10 nm pattern on 2D RRO with 64 binder, we mix scaffold strands, the 10 nm, 14 nm, and 20 nm strands with 64 nt binder, the 10 nm, and 14 nm docking strands with 64 nt binder, the 10 nm docking strands with 64 nt binder along with core staples (positions corresponding to 20 nm docking positions and biotin positions should be excluded beforehand).

Name	Sequence	Note
06_cuboctahedron_147_1-1913-V	ATCACCGTACTTTTTTCAGGAGGTTTTAAAGATTCAATTTTTAAGGGTGAGA	Core staple
06_cuboctahedron_147_1-4390-E	CAGTAACAGTAGTATAGCCCGGAATAGGTGTAGATGAATATA	Core staple
06_cuboctahedron_147_1-4369-E	CGGGAGAAACGCCGTCGAGAGGGTTGATATAACCTTTTACAT	Core staple
06_cuboctahedron_147_1-4348-E	CGCCTGATTGCTCAGTACCAGGCGGATAAGTAATAACGGATT	Core staple
06_cuboctahedron_147_1-4327-E	AAGTTACAAAATTAGGATTAGCGGGGTTTTGCTTTGAATACC	Core staple
06_cuboctahedron_147_1-4306-E	GCGAATTATTCTGAGACTCCTCAAGAGAAGGATCGCGCAGAG	Core staple
06_cuboctahedron_147_1-4285-E	CCTGAGCAAAAAACATGAAAGTATTAAGAGGCATTTCAATTA	Core staple
06_cuboctahedron_147_2-2060-V	ATTTCGGAACCTTTTTTATTATTCTGAGAAGATGATGTTTTTAAACAAACAT	Core staple
06_cuboctahedron_147_2-1609-V	ACTAAAGGAATTTTTTTGCGAATAATGTTTAATTTCATTTTTACTTTAATCA	Core staple
06_cuboctahedron_147_2-1669-E	GTATGGGACAGACGTTAGTAAATGTAACGGGG	Core staple
06_cuboctahedron_147_2-2110-E	TCAGTGCCTACTGGTAATAAGTTTAATTTTCT	Core staple
06_cuboctahedron_147_2-2165-E	GTCATACATGAACAGTTAATGCCCCCTGCCTTTCCAGTAAGC	Core staple
06_cuboctahedron_147_2-2144-E	TACAGGAGTGTTGAGTAACAGTGCCCGTATAGCTTTTGATGA	Core staple
06_cuboctahedron_147_2-1650-E	AACTTTCAACTCTAAAGTTTTGTCGTCTTTCTTTTGCTAAAC	Core staple
06_cuboctahedron_147_2-1629-E	AGTGAGAATACCCTCATAGTTAGCGTAACGAAGTTTCAGCGG	Core staple
06_cuboctahedron_147_2-1619-E	GAAAGGAACACCACAGACAG	Core staple
06_cuboctahedron_147_3-5755-V	AGGGCGAAAAATTTTTCCGTCTATCATAGATTTTCAGTTTTTGTTTAACGTC	Core staple
06_cuboctahedron_147_3-7016-E	CAGTCAAATCGAACGTGGACTCCAACGTCAAAAGGCCGGAGA	Core staple
06_cuboctahedron_147_3-6995-E	GATATTCAACGAACAAGAGTCCACTATTAAAACCATCAATAT	Core staple
06_cuboctahedron_147_3-6974-E	ATAAATTAATGTTGAGTGTTGTTCCAGTTTGCGTTCTAGCTG	Core staple
06_cuboctahedron_147_3-6953-E	TAGCTATTTTAAAAGAATAGCCCGAGATAGGGCCGGAGAGGG	Core staple
06_cuboctahedron_147_3-6932-E	CAAAGGCTATCGGCAAAATCCCTTATAAATCTGAGAGATCTA	Core staple
06_cuboctahedron_147_3-6911-E	CTGAGAGTCTGTTTGATGGTGGTTCCGAAATCAGGTCATTGC	Core staple
06_cuboctahedron_147_4-5902-V	GCCCCAGCAGGTTTTTCGAAAATCCTGGAGCAAACAATTTTTGAGAATCGAT	Core staple
06_cuboctahedron_147_4-5451-V	TTCCTCGTTAGTTTTTAATCAGAGCGACATTTGAGGATTTTTTTTAGAAGTA	Core staple
06_cuboctahedron_147_4-5511-E	CTTAATGCCGCGTAACCACCACACTGATTGCC	Core staple
06_cuboctahedron_147_4-5952-E	CTTCACCGTGAGACGGGCAACAGCCCGCCGCG	Core staple
06_cuboctahedron_147_4-6007-E	TGGGCGCCAGGCAAGCGGTCCACGCTGGTTTGGTTTGCGTAT	Core staple
06_cuboctahedron_147_4-5986-E	TTTTCACCAGCCTGGCCCTGAGAGAGTTGCAGGTGGTTTTTC	Core staple
06_cuboctahedron_147_4-5492-E	GCGCGTACTAGCAAGTGTAGCGGTCACGCTGGCCGCTACAGG	Core staple
06_cuboctahedron_147_4-5471-E	ACGAGCACGTGGAGCGGGCGCTAGGGCGCTGTGGTTGCTTTG	Core staple
06_cuboctahedron_147_4-5461-E	ATAACGTGCTGAAAGCGAAA	Core staple
06_cuboctahedron_147_5-3530-V	TAGAAACCAATTTTTTCAATAATCGGGCGCAGTCTCTTTTTTGAATTTACCG	Core staple
06_cuboctahedron_147_5-4128-V	TTCCCTTAGAATTTTTTCCTTGAAAAAATCGCAAGACTTTTTAAAGAACGCG	Core staple
06_cuboctahedron_147_5-4243-E	AATTAATTACCCATCCTAATTTACGAGCATGCAAGAAAACAA	Core staple
06_cuboctahedron_147_5-4222-E	TCATTTGAATCTGAACAAGAAAAATAATATCATTTAACAATT	Core staple
06_cuboctahedron_147_5-4201-E	ATGGAAACAGTTTATCAACAATAGATAAGTCTACCTTTTTTA	Core staple
06_cuboctahedron_147_5-4180-E	ATATATGTGAAGCTAATGCAGAACGCGCCTGTACATAAATCA	Core staple
06_cuboctahedron_147_5-4159-E	TGCTTCTGTAACGACAATAAACAACATGTTCGTGAATAACCT	Core staple
06_cuboctahedron_147_5-4138-E	TTAATTAATTTAAAGTAATTCTGTCCAGACGAATCGTCGCTA	Core staple
06_cuboctahedron_147_6-1462-V	GACAACAACCATTTTTTCGCCCACGCTTCATGAGGAATTTTTGTTTCCATTA	Core staple
06_cuboctahedron_147_6-1577-E	GTTGAAAATCTAGTAAATTGGGCTTGAGATGAATTTTTTCAC	Core staple
06_cuboctahedron_147_6-1556-E	GGCTCCAAAAGACGAGAAACACCAGAACGAGTCCAAAAAAAA	Core staple
06_cuboctahedron_147_6-1535-E	TTGTATCGGTTCAGTGAATAAGGCTTGCCCTGGAGCCTTTAA	Core staple
06_cuboctahedron_147_6-1514-E	CTTTCGAGGTCAACGTAACAAAGCTGCTCATTTATCAGCTTG	Core staple
06_cuboctahedron_147_6-1493-E	ACAGCTTGATCCGGATATTCATTACCCAAATGAATTTCTTAA	Core staple
06_cuboctahedron_147_6-1472-E	CGCCGACAATCAAGAGTAATCTTGACAAGAAACCGATAGTTG	Core staple
06_cuboctahedron_147_7-6764-V	TGTTAAAATTCTTTTTGCATTAAATTGGCCAACGCGCTTTTTGGGGAGAGGC	Core staple
06_cuboctahedron_147_7-286-V	AGAGTACCTTTTTTTTAATTGCTCCTTTGAGATTTAGTTTTTGAATACCACA	Core staple
06_cuboctahedron_147_7-346-E	TTTTAATTGCCCGAAAGACTTCAAAGCCCCAA	Core staple
06_cuboctahedron_147_7-6814-E	AAACAGGAGGTTGATAATCAGAAAATATCGCG	Core staple
06_cuboctahedron_147_7-6869-E	GTAAAACTAGAAATTGTAAACGTTAATATTTGAACGGTAATC	Core staple
06_cuboctahedron_147_7-6848-E	TATGTACCCCAGATTGTATAAGCAAATATTTCATGTCAATCA	Core staple
06_cuboctahedron_147_7-327-E	GCGAACCAGACATCAAAAAGATTAAGAGGAACGAGCTTCAAA	Core staple
06_cuboctahedron_147_7-306-E	CTCCAACAGGAGTCAGAAGCAAAGCGGATTGCCGGAAGCAAA	Core staple
06_cuboctahedron_147_7-296-E	TCAGGATTAGTGACTATTAT	Core staple
06_cuboctahedron_147_8-6460-V	AACCAGGCAAATTTTTGCGCCATTCGTTCCCAGTCACTTTTTGACGTTGTAA	Core staple
06_cuboctahedron_147_8-6520-E	GTATCGGCTGCCAGTTTGAGGGGAATTCATTG	Core staple
06_cuboctahedron_147_8-493-E	AATCCCCCCGGAATCGTCATAAATCGACGACA	Core staple
06_cuboctahedron_147_8-548-E	AAATGTTTAGGAAAACGAGAATGACCATAAAGGGTAATAGTA	Core staple
06_cuboctahedron_147_8-527-E	TCCAATACTGTCAAATGCTTTAAACAGTTCAACTGGATAGCG	Core staple
06_cuboctahedron_147_8-6501-E	CGCACTCCAGGGCGCATCGTAACCGTGCATCCTCAGGAAGAT	Core staple
06_cuboctahedron_147_8-6480-E	GCACCGCTTCTAGGTCACGTTGGTGTAGATGCCAGCTTTCCG	Core staple
06_cuboctahedron_147_8-6470-E	TGGTGCCGGACGTAATGGGA	Core staple
06_cuboctahedron_147_9-5304-V	TGTAGCAATACTTTTTTTCTTTGATTTGAAATGGATTTTTTTATTTACATTG	Core staple
06_cuboctahedron_147_9-5419-E	GGAGGCCGATTTAGAGCCGTCAATAGATAATGGAGCTAAACA	Core staple
06_cuboctahedron_147_9-5398-E	TAGACAGGAAGGAGCACTAACAACTAATAGATAAAGGGATTT	Core staple
06_cuboctahedron_147_9-5377-E	AATCCTGAGAAGGTTATCTAAAATATCTTTACGGTACGCCAG	Core staple
06_cuboctahedron_147_9-5356-E	AATCAGTGAGACAGTTGAAAGGAATTGAGGAAGTGTTTTTAT	Core staple
06_cuboctahedron_147_9-5335-E	AAAGAGTCTGATCTGGTCAGTTGGCAAATCAGCCACCGAGTA	Core staple

06_cuboctahedron_147_9-5314-E	AATTAACCGTAATATCAAACCCTCAATCAATTCCATCACGCA	Core staple
06_cuboctahedron_147_10-3099-V	AGCGCATTAGATTTTTCGGGAGAATTTAAGAAAAGTATTTTTAGCAGATAGC	Core staple
06_cuboctahedron_147_10-3214-E	GTTACAAAATCAGAATCAAGTTTGCCTTTAGCTAATTTGCCA	Core staple
06_cuboctahedron_147_10-3193-E	TTATTTATCCCAGCACCGTAATCAGTAGCGAAAACAGCCATA	Core staple
06_cuboctahedron_147_10-3172-E	AGAAACGATTCACCAATGAAACCATCGATAGCAATCCAAATA	Core staple
06_cuboctahedron_147_10-3151-E	GTCAAAAATGCATTAGCAAGGCCGGAAACGTTTTTGTTTAAC	Core staple
06_cuboctahedron_147_10-3130-E	CTTTACAGAGAATCACCAGTAGCACCATTACAAAATAGCAGC	Core staple
06_cuboctahedron_147_10-3109-E	AAAACAGGGATTTGGGAATTAGAGCCAGCAAAGAATAACATA	Core staple
06_cuboctahedron_147_11-2785-V	AAAAGAAACGCTTTTTAAAGACACCACACCGTCACCGTTTTTACTTGAGCCA	Core staple
06_cuboctahedron_147_11-2845-E	TCCTTATTCAAAAGAACTGGCATGGCCAGCTG	Core staple
06_cuboctahedron_147_11-6373-E	GCGAAAGGGCCTCTTCGCTATTACATTAAGAC	Core staple
06_cuboctahedron_147_11-6428-E	GCGCAACTGTAAGTTGGGTAACGCCAGGGTTCCATTCAGGCT	Core staple
06_cuboctahedron_147_11-6407-E	ATCGGTGCGGGGGATGTGCTGCAAGGCGATTTGGGAAGGGCG	Core staple
06_cuboctahedron_147_11-2826-E	TAGCAAACGTACGCAATAATAACGGAATACCACGCAGTATGT	Core staple
06_cuboctahedron_147_11-2805-E	ACATAAAGGTTACCAGAAGGAAACCGAGGAAAGAAAATACAT	Core staple
06_cuboctahedron_147_11-2795-E	GGCAACATATCGAACAAAGT	Core staple
06_cuboctahedron_147_12-1881-E	CCTCAGAACCTAGTAGTAGCATTTGTGTAGGAGTACCGCCAC	Core staple
06_cuboctahedron_147_12-1860-E	AACCGCCACCAGGTGGCATCAATTCTACTAAGCCACCCTCAG	Core staple
06_cuboctahedron_147_12-1839-E	CACCCTCATTCATTTGGGGCGCGAGCTGAAACTCAGAGCCAC	Core staple
06_cuboctahedron_147_12-1818-E	CAAGCCCAATTAACCTGTTTAGCTATATTTTTTCAGGGATAG	Core staple
06_cuboctahedron_147_12-1797-E	TACCGTAACAATACATTTCGCAAATGGTCAAAGGAACCCATG	Core staple
06_cuboctahedron_147_12-1776-E	CACCAGTACATAGATTTAGTTTGACCATTAGCTGAGTTTCGT	Core staple
06_cuboctahedron_147_13-139-V	ATTCCCAATTCTTTTTTGCGAACGAGAACTACAACGCTTTTTCTGTAGCATT	Core staple
06_cuboctahedron_147_13-832-E	CTTATGCGATTTTCATTCCATATAACAGTTGTTGTGAATTAC	Core staple
06_cuboctahedron_147_13-811-E	GCTCATTATACTAAAGTACGGTGTCTGGAAGTTTAAGAACTG	Core staple
06_cuboctahedron_147_13-790-E	GTTGGGAAGACAACATGTTTTAAATATGCAACCAGTCAGGAC	Core staple
06_cuboctahedron_147_13-769-E	TAATAAAACGGCTGAATATAATGCTGTAGCTAAAATCTACGT	Core staple
06_cuboctahedron_147_13-748-E	CAACATTATTCGGATGGCTTAGAGCTTAATTAACTAACGGAA	Core staple
06_cuboctahedron_147_13-727-E	GATTCATCAGTTTGATAAGAGGTCATTTTTGACAGGTAGAAA	Core staple
06_cuboctahedron_147_14-5723-E	CACTACGTGAAACAGAAATAAAGAAATTGCGGGGCGATGGCC	Core staple
06_cuboctahedron_147_14-5702-E	AATCAAGTTTATCAAAATTATTTGCACGTAAACCATCACCCA	Core staple
06_cuboctahedron_147_14-5681-E	GGTGCCGTAAGGAAGGGTTAGAACCTACCATTTTGGGGTCGA	Core staple
06_cuboctahedron_147_14-5660-E	GGAACCCTAATGGATTATACTTCTGAATAATAGCACTAAATC	Core staple
06_cuboctahedron_147_14-5639-E	GATTTAGAGCATCAATATAATCCTGATTGTTAGGGAGCCCCC	Core staple
06_cuboctahedron_147_14-5618-E	AGCCGGCGAAATTATCAGATGATGGCAATTCTTGACGGGGAA	Core staple
06_cuboctahedron_147_15-4569-V	GGAATTATCATTTTTTCATATTCCTGCGTGGCGAGAATTTTTAGGAAGGGAA	Core staple
06_cuboctahedron_147_15-4031-E	CTTTTTAAAAAATCATAGGTCTGATTTTAAAA	Core staple
06_cuboctahedron_147_15-4619-E	GTTTGAGTGCCCGAACGTTATTAAGAGACTAC	Core staple
06_cuboctahedron_147_15-4674-E	AAACAATTCGAAAGAAACCACCAGAAGGAGCTTAGACTTTAC	Core staple
06_cuboctahedron_147_15-4653-E	TAAATCCTTTAACATTATCATTTTGCGGAACACAACTCGTAT	Core staple
06_cuboctahedron_147_15-4012-E	GGTTGGGTTAAGAGTCAATAGTGAATTTATCCCTCCGGCTTA	Core staple
06_cuboctahedron_147_15-3991-E	GTAAATGCTGGCTTAGATTAAGACGCTGAGATATAACTATAT	Core staple
06_cuboctahedron_147_15-3981-E	ATGCAAATCCCATAGCGATA	Core staple
06_cuboctahedron_147_16-3498-E	TATCATTCCATCATTAAAGCCAGAATGGAAACTGTCTTTCCT	Core staple
06_cuboctahedron_147_16-3477-E	TAAACCAAGTTATTCACAAACAAATAAATCCAGAACGGGTAT	Core staple
06_cuboctahedron_147_16-3456-E	CGAGAACAAGAGGTCAGACGATTGGCCTTGAACCGCACTCAT	Core staple
06_cuboctahedron_147_16-3435-E	TATTTTCATCGCATTGACAGGAGGTTGAGGCCAAGCCGTTTT	Core staple
06_cuboctahedron_147_16-3414-E	TACCGCGCCCCCACCACCAGAGCCGCCGCCAGTAGGAATCAT	Core staple
06_cuboctahedron_147_16-3393-E	AATCAGATATCCCTCAGAGCCGCCACCAGAAAATAGCAAGCA	Core staple
06_cuboctahedron_147_17-2344-V	CCGCCACCCTCTTTTTAGAGCCACCAAGAAGGCTTATTTTTTCCGGTATTCT	Core staple
06_cuboctahedron_147_17-1365-E	AAAGACAGGCGGGATCGTCACCCTATCACCGG	Core staple
06_cuboctahedron_147_17-2394-E	AACCAGAGTCTTTTCATAATCAAACAGCAGCG	Core staple
06_cuboctahedron_147_17-2449-E	TCGGTCATAGTCAGAGCCGCCACCCTCAGAACATCGGCATTT	Core staple
06_cuboctahedron_147_17-2428-E	GCGTTTGCCACCACCACCGGAACCGCCTCCCCCCCCTTATTA	Core staple
06_cuboctahedron_147_17-1346-E	GGGTAGCAACAGGGAGTTAAAGGCCGCTTTTCATCGGAACGA	Core staple
06_cuboctahedron_147_17-1325-E	CTTTGAGGACTATTCGGTCGCTGAGGCTTGCGGCTACAGAGG	Core staple
06_cuboctahedron_147_17-1315-E	TAAAGACTTTATAACCGATA	Core staple
06_cuboctahedron_147_18-6732-E	AGCTCATTTTTGCCAGCTGCATTAATGAATCTTTGTTAAATC	Core staple
06_cuboctahedron_147_18-6711-E	GAACGCCATCTTCCAGTCGGGAAACCTGTCGTTAACCAATAG	Core staple
06_cuboctahedron_147_18-6690-E	GCGTCTGGCCGCGTTGCGCTCACTGCCCGCTAAAAATAATTC	Core staple
06_cuboctahedron_147_18-6669-E	AGCTTTCATCGTGAGCTAACTCACATTAATTTTCCTGTAGCC	Core staple
06_cuboctahedron_147_18-6648-E	TGAGCGAGTAAAAGCCTGGGGTGCCTAATGAAACATTAAATG	Core staple
06_cuboctahedron_147_18-6627-E	GATTCTCCGTCGAGCCGGAAGCATAAAGTGTACAACCCGTCG	Core staple
06_cuboctahedron_147_19-6186-V	CTCACAATTCCTTTTTACACAACATAGGGAACAAACGTTTTTGCGGATTGAC	Core staple
06_cuboctahedron_147_19-5207-E	CCAGCCATGTAATATCCAGAACAAACCGAGCT	Core staple
06_cuboctahedron_147_19-6236-E	CGAATTCGTAGAGGATCCCCGGGTTATTACCG	Core staple
06_cuboctahedron_147_19-6291-E	GTGCCAAGCTCCTGTGTGAAATTGTTATCCGAACGACGGCCA	Core staple
06_cuboctahedron_147_19-6270-E	AGGTCGACTCTAATCATGGTCATAGCTGTTTTGCATGCCTGC	Core staple
06_cuboctahedron_147_19-5188-E	AAACGCTCATCTCAAACTATCGGCCTTGCTGTGCAACAGGAA	Core staple
06_cuboctahedron_147_19-5167-E	CATTTTGACGTCACTTGCCTGAGTAGAAGAAGGAAATACCTA	Core staple
06_cuboctahedron_147_19-5157-E	CTCAATCGTCAGTAATAACA	Core staple

06_cuboctahedron_147_20-3246-V	TAACGAGCGTCTTTTTTTTCCAGAGCCGTCAGACTGTTTTTTAGCGCGTTTT	Core staple
06_cuboctahedron_147_20-3737-E	ATATTTAAAGTAGGGCTTAATTGATCAAGATT	Core staple
06_cuboctahedron_147_20-3296-E	AGTTGCTAGTTTTGAAGCCTTAAAGAATCGCC	Core staple
06_cuboctahedron_147_20-3351-E	GCGTTTTAGCTATCCTGAATCTTACCAACGCAAGAACGCGAG	Core staple
06_cuboctahedron_147_20-3330-E	CTTGCGGGAGTTTTGCACCCAGCTACAATTTGAACCTCCCGA	Core staple
06_cuboctahedron_147_20-3718-E	TGTAATTTAGAGTATAAAGCCAACGCTCAACCAACGCCAACA	Core staple
06_cuboctahedron_147_20-3697-E	TTCGAGCCAGTGCGTTATACAAATTCTTACCGCAGAGGCATT	Core staple
06_cuboctahedron_147_20-3687-E	TAATAAGAGATAGTATCATA	Core staple
06_cuboctahedron_147_21-3834-V	AATTACTAGAATTTTTAAAGCCTGTTATATAAAGTACTTTTTCGACAAAAGG	Core staple
06_cuboctahedron_147_21-4913-E	GTATTAACGATAAAACAGAGGTGATTGAAATA	Core staple
06_cuboctahedron_147_21-3884-E	CCGACCGTACCTAAATTTAATGGTGGCGGTCA	Core staple
06_cuboctahedron_147_21-3939-E	TCAAATATATAAGAATAAACACCGGAATCATAGAAAACTTTT	Core staple
06_cuboctahedron_147_21-3918-E	TCATCTTCTGGTGATAAATAAGGCGTTAAATTTTAGTTAATT	Core staple
06_cuboctahedron_147_21-4894-E	CAGTGCCACGCCGAACGAACCACCAGCAGAAACCGCCTGCAA	Core staple
06_cuboctahedron_147_21-4873-E	CAGCAAATGAAAAACATCGCCATTAAAAATACTGAGAGCCAG	Core staple
06_cuboctahedron_147_21-4863-E	AAAATCTAAATGATAGCCCT	Core staple
06_cuboctahedron_147_22-5010-V	ATTAGTCTTTATTTTTATGCGCGAACGCATCACCTTGTTTTTCTGAACCTCA	Core staple
06_cuboctahedron_147_22-3002-E	GCCCAATAGATAACCCACAAGAATAACCCTTC	Core staple
06_cuboctahedron_147_22-5060-E	TGACCTGATGGCCAACAGAGATAGTGAGTTAA	Core staple
06_cuboctahedron_147_22-5115-E	AGTCACACGAAGACAATATTTTTGAATGGCTGCAGATTCACC	Core staple
06_cuboctahedron_147_22-5094-E	AGGGACATTCAAGCGTAAGAATACGTGGCACCCAGTAATAAA	Core staple
06_cuboctahedron_147_22-2983-E	AAACAATGAAATTGAGCGCTAATATCAGAGAATAAGAGCAAG	Core staple
06_cuboctahedron_147_22-2962-E	TATCTTACCGCCTGAACAAAGTCAGAGGGTAATAGCAATAGC	Core staple
06_cuboctahedron_147_22-2952-E	AAGCCCTTTTAACTGAACAC	Core staple
06_cuboctahedron_147_23-580-V	CAAAAGAAGTTTTTTTTTGCCAGAGGTCAAAAATCAGTTTTTGTCTTTACCC	Core staple
06_cuboctahedron_147_23-1071-E	TAAGGGAAGAACGAGGCGCAGACGCTATCATA	Core staple
06_cuboctahedron_147_23-630-E	ACCCTCGTCATAGTAAGAGCAACAGTCAATCA	Core staple
06_cuboctahedron_147_23-685-E	CAGATACATACAAAATAGCGAGAGGCTTTTGTTCAACTAATG	Core staple
06_cuboctahedron_147_23-664-E	AATTACGAGGTTACCAGACGACGATAAAAACACGCCAAAAGG	Core staple
06_cuboctahedron_147_23-1052-E	AACTTTGAAACTGCTCCATGTTACTTAGCCGCCGAACTGACC	Core staple
06_cuboctahedron_147_23-1031-E	AACGGTGTACAATTGTGTCGAAATCCGCGACGAGGACAGATG	Core staple
06_cuboctahedron_147_23-1021-E	AGACCAGGCGTCGCCTGATA	Core staple
06_cuboctahedron_147_24-1168-V	GTACAACGGAGTTTTTATTTGTATCACATAGGCTGGCTTTTTTGACCTTCAT	Core staple
06_cuboctahedron_147_24-2688-E	CAACCGATCGCCAAAGACAAAAGGAGAATACA	Core staple
06_cuboctahedron_147_24-1218-E	CTAAAACAAAACGAAAGAGGCAAAGCGACATT	Core staple
06_cuboctahedron_147_24-1273-E	TACGTAATGCTTATACCAAGCGCGAAACAAAAACGGGTAAAA	Core staple
06_cuboctahedron_147_24-1252-E	CACCAACCTACTCATCTTTGACCCCCAGCGACACTACGAAGG	Core staple
06_cuboctahedron_147_24-2669-E	AGGTAAATATAAAATTCATATGGTTTACCAGTGAGGGAGGGA	Core staple
06_cuboctahedron_147_24-2648-E	ATTCATTAAATTATTTTGTCACAATCAATAGTGACGGAAATT	Core staple
06_cuboctahedron_147_24-2638-E	GGTGAATTATCGGAATAAGT	Core staple
06_cuboctahedron_147_5-4128-Vertex-3T-R1	TTCCCTTAGAATTTTTTCCTTGAAAAAATCGCAAGACTTTTTAAAGAACGCGTTTCCTCCTCCTCCTCCTCCT	Vertex 1a docking strand
06_cuboctahedron_147_21-3834-Vertex-3T-R1	AATTACTAGAATTTTTAAAGCCTGTTATATAAAGTACTTTTTCGACAAAAGGTTTCCTCCTCCTCCTCCTCCT	Vertex 1b docking strand
06_cuboctahedron_147_17-2344-Vertex-3T-R1	CCGCCACCCTCTTTTTAGAGCCACCAAGAAGGCTTATTTTTTCCGGTATTCTTTTCCTCCTCCTCCTCCTCCT	Vertex 2a docking strand
06_cuboctahedron_147_20-3246-Vertex-3T-R1	TAACGAGCGTCTTTTTTTTCCAGAGCCGTCAGACTGTTTTTTAGCGCGTTTTTTTCCTCCTCCTCCTCCTCCT	Vertex 2b docking strand
06_cuboctahedron_147_10-3099-Vertex-3T-R1	AGCGCATTAGATTTTTCGGGAGAATTTAAGAAAAGTATTTTTAGCAGATAGCTTTCCTCCTCCTCCTCCTCCT	Vertex 3a docking strand
06_cuboctahedron_147_11-2785-Vertex-3T-R1	AAAAGAAACGCTTTTTAAAGACACCACACCGTCACCGTTTTTACTTGAGCCATTTCCTCCTCCTCCTCCTCCT	Vertex 3b docking strand
06_cuboctahedron_147_9-5304-Vertex-3T-R1	TGTAGCAATACTTTTTTTCTTTGATTTGAAATGGATTTTTTTATTTACATTGTTTCCTCCTCCTCCTCCTCCT	Vertex 4a docking strand
06_cuboctahedron_147_22-5010-Vertex-3T-R1	ATTAGTCTTTATTTTTATGCGCGAACGCATCACCTTGTTTTTCTGAACCTCATTTCCTCCTCCTCCTCCTCCT	Vertex 4b docking strand
06_cuboctahedron_147_1-1913-Vertex-3T-R1	ATCACCGTACTTTTTTCAGGAGGTTTTAAAGATTCAATTTTTAAGGGTGAGATTTCCTCCTCCTCCTCCTCCT	Vertex 5a docking strand
06_cuboctahedron_147_3-5755-Vertex-3T-R1	AGGGCGAAAAATTTTTCCGTCTATCATAGATTTTCAGTTTTTGTTTAACGTCTTTCCTCCTCCTCCTCCTCCT	Vertex 5b docking strand
06_cuboctahedron_147_2-1609-Vertex-3T-R1	ACTAAAGGAATTTTTTTGCGAATAATGTTTAATTTCATTTTTACTTTAATCATTTCCTCCTCCTCCTCCTCCT	Vertex 6a docking strand
06_cuboctahedron_147_13-139-Vertex-3T-R1	ATTCCCAATTCTTTTTTGCGAACGAGAACTACAACGCTTTTTCTGTAGCATTTTTCCTCCTCCTCCTCCTCCT	Vertex 6b docking strand
06_cuboctahedron_147_7-286-Vertex-3T-R1	AGAGTACCTTTTTTTTTAATTGCTCCTTTGAGATTTAGTTTTTGAATACCACATTTCCTCCTCCTCCTCCTCCT	Vertex 7a docking strand
06_cuboctahedron_147_23-580-Vertex-3TR1	CAAAAGAAGTTTTTTTTTGCCAGAGGTCAAAAATCAGTTTTTGTCTTTACCCTTTCCTCCTCCTCCTCCTCCT	Vertex 7b docking strand
06_cuboctahedron_147_4-5902-Vertex-3T-R1	GCCCCAGCAGGTTTTTCGAAAATCCTGGAGCAAACAATTTTTGAGAATCGATTTTCCTCCTCCTCCTCCTCCT	Vertex 8a docking strand
06_cuboctahedron_147_7-6764-Vertex-3T-R1	TGTTAAAATTCTTTTTGCATTAAATTGGCCAACGCGCTTTTTGGGGAGAGGCTTTCCTCCTCCTCCTCCTCCT	Vertex 8b docking strand
06_cuboctahedron_147_4-5451-Vertex-3T-R1	TTCCTCGTTAGTTTTTAATCAGAGCGACATTTGAGGATTTTTTTTAGAAGTATTTCCTCCTCCTCCTCCTCCT	Vertex 9a docking strand
06_cuboctahedron_147_15-4569-Vertex-3T-R1	GGAATTATCATTTTTTCATATTCCTGCGTGGCGAGAATTTTTAGGAAGGGAATTTCCTCCTCCTCCTCCTCCT	Vertex 9b docking strand
06_cuboctahedron_147_2-2060-Vertex-3T-R1	ATTTCGGAACCTTTTTTATTATTCTGAGAAGATGATGTTTTTAAACAAACATTTCCTCCTCCTCCTCCTCCT	Vertex 10a docking strand
06_cuboctahedron_147_5-3530-Vertex-3T-R1	TAGAAACCAATTTTTTCAATAATCGGGCGCAGTCTCTTTTTTGAATTTACCGTTTCCTCCTCCTCCTCCTCCT	Vertex 10b docking strand
06_cuboctahedron_147_6-1462-Vertex-3T-R1	GACAACAACCATTTTTTCGCCCACGCTTCATGAGGAATTTTTGTTTCCATTATTTCCTCCTCCTCCTCCTCCT	Vertex 11a docking strand
06_cuboctahedron_147_24-1168-Vertex-3T-R1	GTACAACGGAGTTTTTATTTGTATCACATAGGCTGGCTTTTTTGACCTTCATTTCCTCCTCCTCCTCCTCCT	Vertex 11b docking strand
06_cuboctahedron_147_8-6460-Vertex-3T-R1	AACCAGGCAAATTTTTGCGCCATTCGTTCCCAGTCACTTTTTGACGTTGTAATTTCCTCCTCCTCCTCCTCCT	Vertex 12a docking strand
06_cuboctahedron_147_19-6186-Vertex-3T-R1	CTCACAATTCCTTTTTACACAACATAGGGAACAAACGTTTTTGCGGATTGACTTTCCTCCTCCTCCTCCTCCT	Vertex 12b docking strand
06_cuboctahedron_147_21-3884-Edge-3T-R1	CCGACCGTACCTAAATTTAATGGTGGCGGTCATTTCCTCCTCCTCCTCCTCCT	Edge docking strand
06_cuboctahedron_147_20-3296-Edge-3T-R1	AGTTGCTAGTTTTGAAGCCTTAAAGAATCGCCTTTCCTCCTCCTCCTCCTCCT	Edge docking strand
06_cuboctahedron_147_10-3172-Edge-3T-R1	AGAAACGATTCACCAATGAAACCATCGATAGCAATCCAAATATTTCCTCCTCCTCCTCCTCCT	Edge docking strand
06_cuboctahedron_147_22-5060-Edge-3T-R1	TGACCTGATGGCCAACAGAGATAGTGAGTTAATTTCCTCCTCCTCCTCCTCCT	Edge docking strand
06_cuboctahedron_147_3-6974-Edge-3T-R1	ATAAATTAATGTTGAGTGTTGTTCCAGTTTGCGTTCTAGCTGTTTCCTCCTCCTCCTCCTCCT	Edge docking strand
06_cuboctahedron_147_12-1839-Edge-3T-R1	CACCCTCATTCATTTGGGGCGCGAGCTGAAACTCAGAGCCACTTTCCTCCTCCTCCTCCTCCT	Edge docking strand
06_cuboctahedron_147_13-790-Edge-3T-R1	GTTGGGAAGACAACATGTTTTAAATATGCAACCAGTCAGGACTTTCCTCCTCCTCCTCCTCCT	Edge docking strand

06_cuboctahedron_147_7-6814-Edge-3T-R1	AAACAGGAGGTTGATAATCAGAAAATATCGCGTTTCCTCCTCCTCCTCCT	Edge docking strand
biotin-06_cuboctahedron_147_21-3918-Edge	/5Biosg/TCATCTTCTGGTGATAAATAAGGCGTTAAATTTTAGTTAATT	Biotinylated staple
biotin-06_cuboctahedron_147_21-4894-Edge	/5Biosg/CAGTGCCACGCCGAACGAACCACCAGCAGAAACCGCCTGCAA	Biotinylated staple
biotin-06_cuboctahedron_147_22-5094-Edge	/5Biosg/AGGGACATTCAAGCGTAAGAATACGTGGCACCCAGTAATAAA	Biotinylated staple
biotin-06_cuboctahedron_147_22-2983-Edge	/5Biosg/AAACAATGAAATTGAGCGCTAATATCAGAGAATAAGAGCAAG	Biotinylated staple
biotin-06_cuboctahedron_147_10-3130-Edge	/5Biosg/CTTTACAGAGAATCACCAGTAGCACCATTACAAAATAGCAGC	Biotinylated staple
biotin-06_cuboctahedron_147_10-3193-Edge	/5Biosg/TTATTTATCCCAGCACCGTAATCAGTAGCGAAAACAGCCATA	Biotinylated staple
biotin-06_cuboctahedron_147_20-3330-Edge	/5Biosg/CTTGCGGGAGTTTTGCACCCAGCTACAATTTGAACCTCCCGA	Biotinylated staple
biotin-06_cuboctahedron_147_20-3697-Edge	/5Biosg/TTCGAGCCAGTGCGTTATACAAATTCTTACCGCAGAGGCATT	Biotinylated staple

Table S2 **3D wireframe cuboctahedron DNA origami strands.** Scaffold is MP13mp18. To make a pattern on 3D wireframe cuboctahedron DNA origami, we mix scaffold strands, biotinylated strands, core staple strands (positions corresponding to the pattern docking positions and biotin positions should be excluded beforehand) along with the corresponding docking strands that make up the pattern.

Letters	**Binary**	**Letters**	**Binary**
A	1000001	T	1010100
B	1000010	U	1010101
C	1000011	V	1010110
D	1000100	W	1010111
E	1000101	X	1011000
F	1000110	Y	1011001
G	1000111	Z	1011010
H	1001000	Space	100000
I	1001001	1	110001
J	1001010	2	110010
K	1001011	3	110011
L	1001100	4	110100
M	1001101	5	110101
N	1001110	6	110110
O	1001111	7	110111
P	1010000	8	111000
Q	1010001	9	111001
R	1010010	0	110000
S	1010011		

Table S3 Letters to binary and number to binary. In our demonstration, the last six digits of the binary encoding are assigned to the alphabets while the last four digits are allocated to the numbers.

Strands	**Concentration**
M13mp18	20 nM
Core staple (positions corresponding to the pattern docking positions and biotin positions should be excluded beforehand)	200 nM/strand
Biotinylated staple	1000 nM/strand
Corresponding docking strands	1250 nM/strand
TAE MgCl2 buffer	1×

Table S4 Mixing concentrations for all experiments in the main text figures. In our demonstration, the last six digits of the binary encoding are assigned to the alphabets while the last four digits are allocated to the numbers.

Imaging Parameters	**NSF 2D dataset**	**20 nm RRO 32 nt binder**	**20 nm RRO 64 nt binder**	**14 nm RRO 64 nt binder**	**10 nm RRO 64 nt binder**	**ASU one-redundancy 2D dataset**	**ASU two-redundancy 2D dataset**	**0407 3D dataset**
DNA origami concentration	1 nm, no fiduciary drift correction markers	1 nm, no fiduciary drift correction markers	1 nM, no fiduciary drift correction markers	1 nM, no fiduciary drift correction markers	1 nm, no fiduciary drift correction markers	1 nm, no fiduciary drift correction markers	1 nm, no fiduciary drift correction markers	1.5 nM with 0.5 nM of 20 nm RRO for fiduciary drift correction markers
Imager concentration	5 nM	5 nM	5 nM	5 nM	2 nM	1 nM	1 nM	1 nM
PCA, PCD, Trolox concentration	1.25X PCA, 1× PCD and 1× Trolox	1.25X PCA, 1× PCD and 1× Trolox	1.25X PCA, 1× PCD and 1× Trolox	1.25X PCA, 1× PCD and 1× Trolox	1.25X PCA, 1× PCD and 1× Trolox	1.25X PCA, 1× PCD and 1× Trolox	1.25X PCA, 1× PCD and 1× Trolox	1.25X PCA, 1× PCD and 1× Trolox
Camera exposure time	50 ms	50 ms	50 ms	50 ms	50 ms	50 ms	50 ms	50 ms
Laser power density	800 W/cm^2	800 W/cm^2	800 W/cm^2	800 W/cm^2	1250 W/cm^2	1250 W/cm^2	1250 W/cm^2	1250 W/cm^2
No. of frames	15,000	15,000	15,000	30,000	90,000	30,000	30,000	43,510
TIRF	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3D lens	No	No	No	No	No	No	No	Yes

Table S5 DNA-PAINT super-resolution imaging parameters for each experiment

Parameters	**Values**
Box side length	7 for 2D, and 9 for 3D
Min. Net Gradient	15,000 for 2D, and 10,000 for 3D (filtering can be done later using Picasso Filter module
EM Gain	1
Baseline	100
Sensitivity	0.46
Quantum efficiency (at Cy3B emission)	0.82
Pixel size	117 nm
Method	MLE, integrated Gaussian for 2D, and LQ, Gaussian for 3D
3D via Astigmatism	Empty for 2D, and Use a calibration file for 3D

Table S6 Picasso localize module parameters. The parameters are based on the Hamamatsu ORCA-Flash4.0 V3 digital sCMOS camera.

Parameter	**NSF 2D dataset**	**ASU one-redundancy 2D dataset**	**ASU two-redundancy 2D dataset**	**0407 3D dataset**
N	3	9	9	0
M	13	49	49	13
T_I	0.15	0.2	0.2	95%
T_S	0.5	0.7	0.8	Not applicable
W_O	1.5	1.5	1.5	1
Alignment	Rough	Differential evolution	Differential evolution	Steps of translation followed by rotation with respect to z-axis

Table S7 Parameter selection. It is done by empirically finding good values for N and M. If origami with higher numbers of binding sites are imaged, we may need a higher value of M to account for false positives. T_I and T_S were selected through grid search. W_O was empirically selected. The best method for alignment for each dataset was empirically selected.

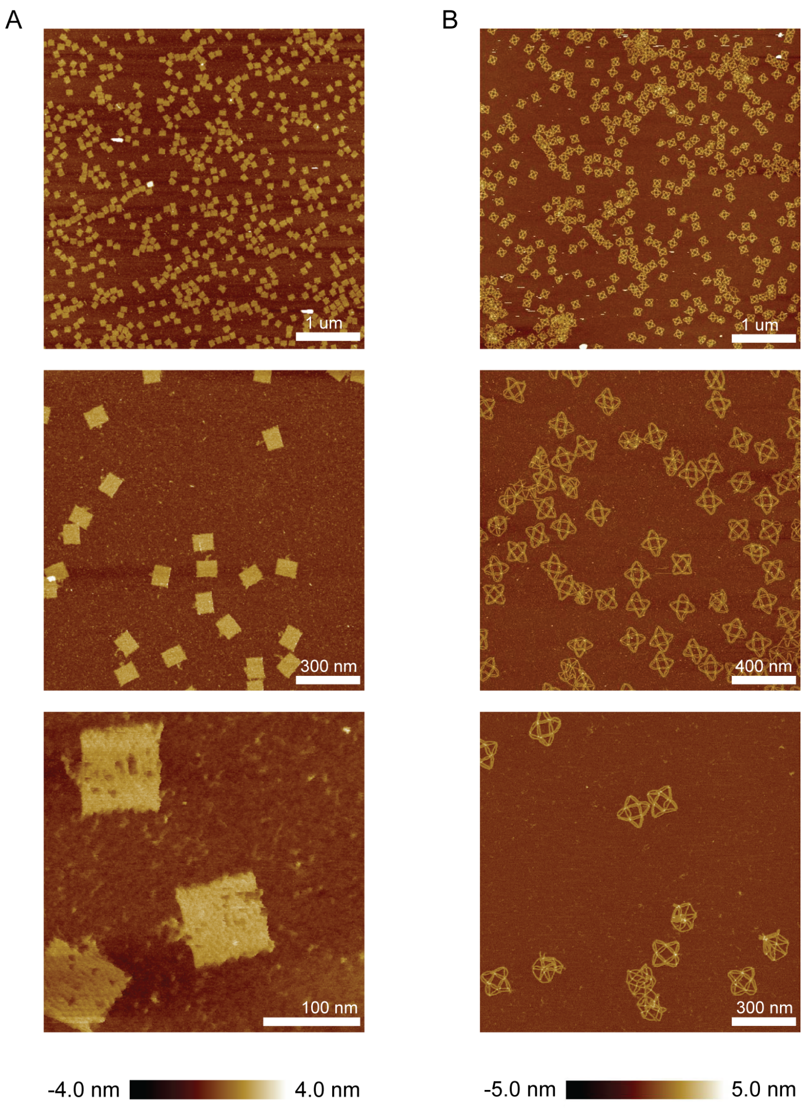


Fig. S1 **Additional AFM images of 2D RRO and 3D wireframe cuboctahedron DNA origami.**(**A**) AFM images of 2D RRO with varying fields of view. (**B**) AFM images of 3D cuboctahedron DNA origami with varying fields of view.

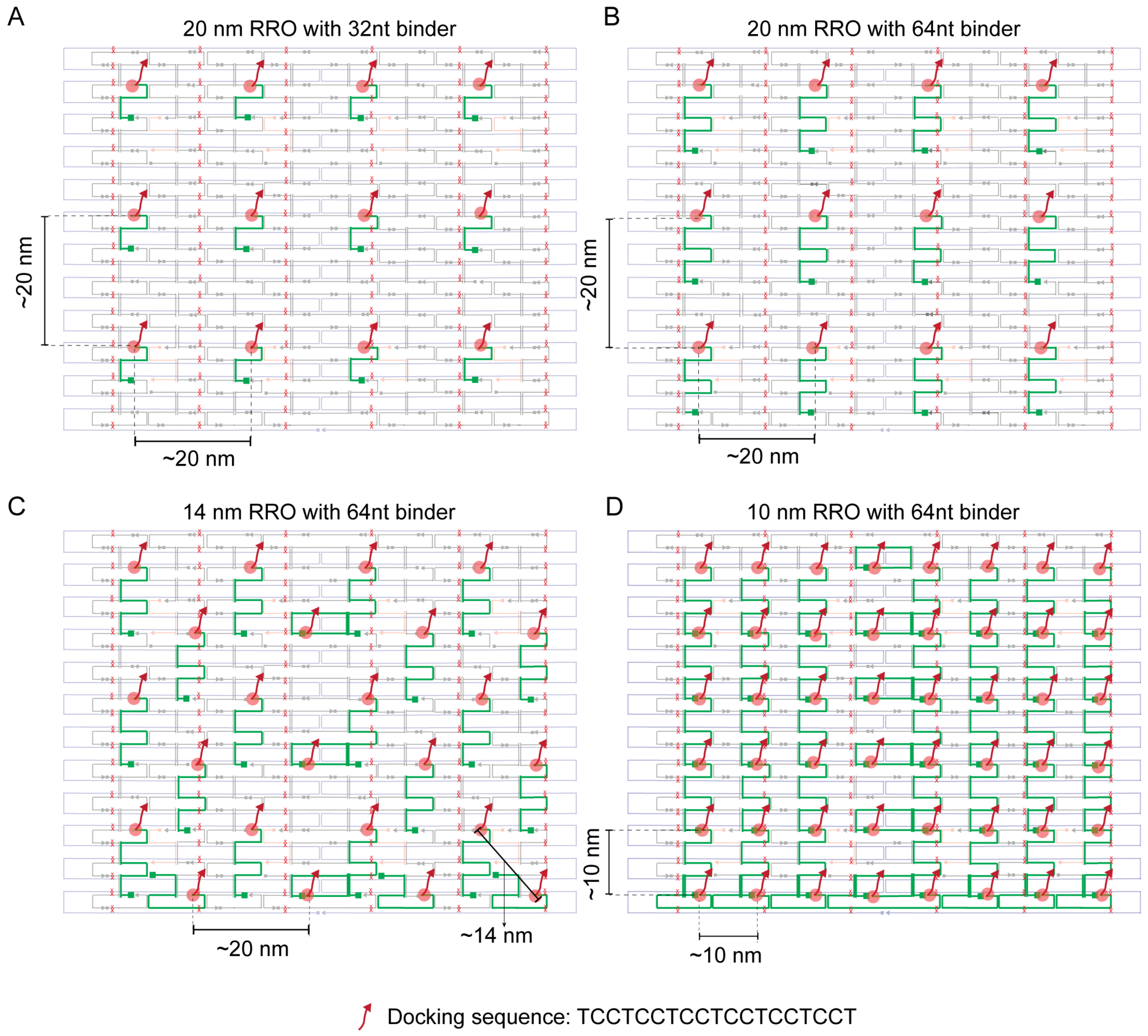

A
20 nm RRO with 32nt binder
~20 nm
~20 nm
B
20 nm RRO with 64nt binder
~20 nm
~20 nm
C
14 nm RRO with 64nt binder
~20 nm
~14 nm
D
10 nm RRO with 64nt binder
~10 nm
~10 nm
Docking sequence: TCCTCCTCCTCCTCCTCCT

Fig. S2 **2D RRO cadnano design showing scaffold routing and staple strands interlacing** (**A**) Map of 20 nm 2D RRO with 32 nt binder and 20 nm separation between imager binding locations. (**B**) Map of 20 nm 2D RRO with 64nt binder and 20 nm separation between imager binding locations. (**C**) Map of 14 nm 2D RRO with long 64 nt binder and 14 nm and 20 nm separation between imager binding locations. (**D**) Map of 10 nm 2D RRO with 64 nt binder and 10 nm separation between imager binding locations.

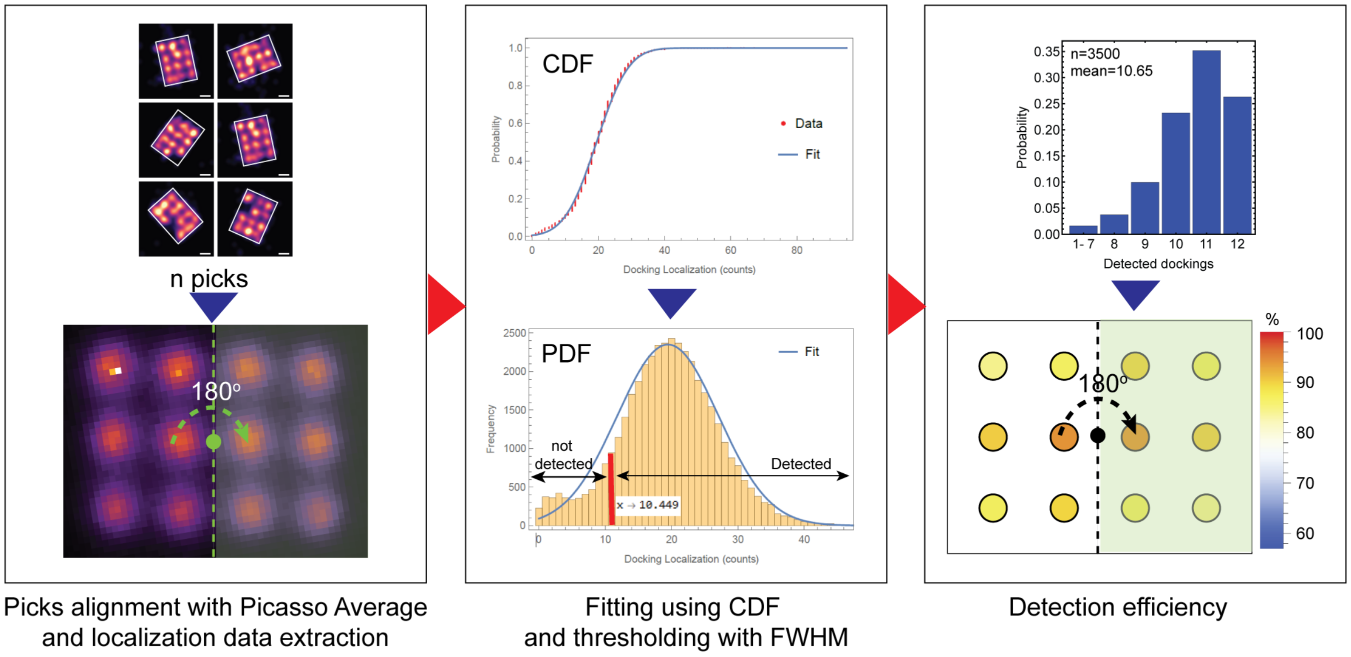


Fig. S3 **Detection efficiency analysis procedure in 20 nm RRO with 32 nt binder.** Procedure for docking detection efficiency of 20 nm RRO with 32 nt binder. Left-most panel shows Picasso Render picking process (top) being input to Picasso average module to align the picks (bottom). Middle panel is the incorporation distribution of localization for each docking in all picks (bottom) fitted by using cumulative distribution function (CDF) (top) and thresholded by using full width half maximum (FWHM). Rightmost panel shows localization distribution after thresholding and each docking detection efficiency from all picks.

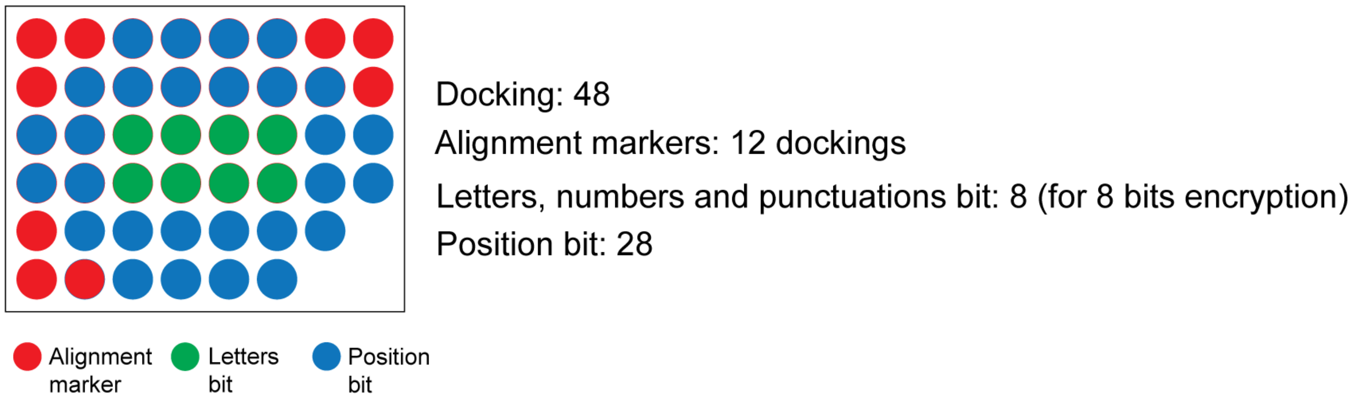

Docking: 48
Alignment markers: 12 dockings
Letters, numbers and punctuations bit: 8 (for 8 bits encryption)
Position bit: 28
Alignment marker
Letters bit
Position bit

Fig. S4 **Theoretical design of 10 nm resolution encryption pattern resulting in 2^{28} combinations of numbers, letters and punctuation marks forming texts assuming 100% incorporation efficiency.** Design of 10 nm encryption showing the alignment marker with 12 dockings which breaks symmetry of design by including only 9 dockings in design (red), letters bit with 8 dockings (green), and position bit with 28 dockings (blue).

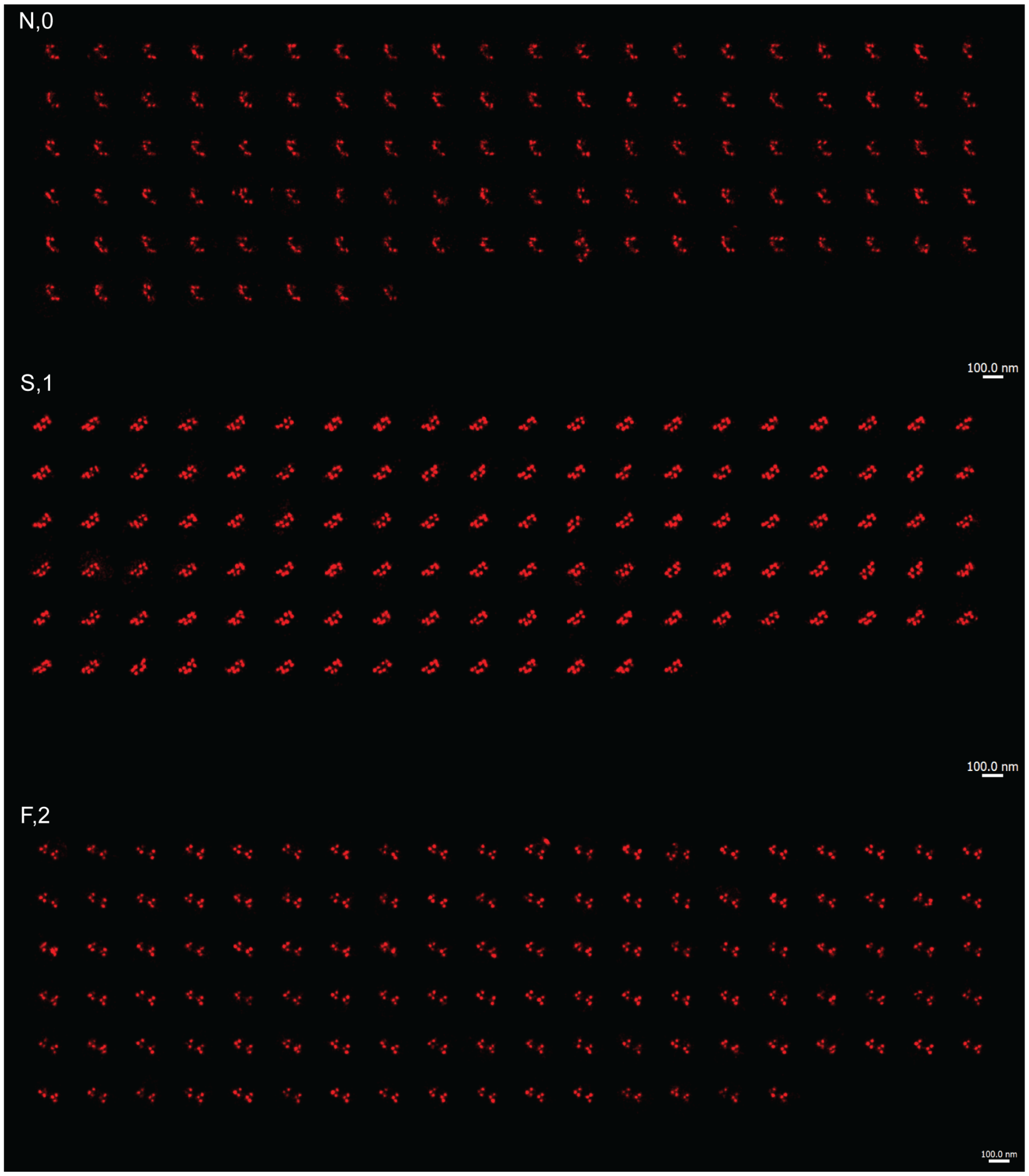


Fig. S5 **All picks in the analyzed "NSF" dataset.** Full data set of 20 nm encrypted "NSF" following Picasso Average alignment and Picasso render unfolding with a 100 nm scale bar.

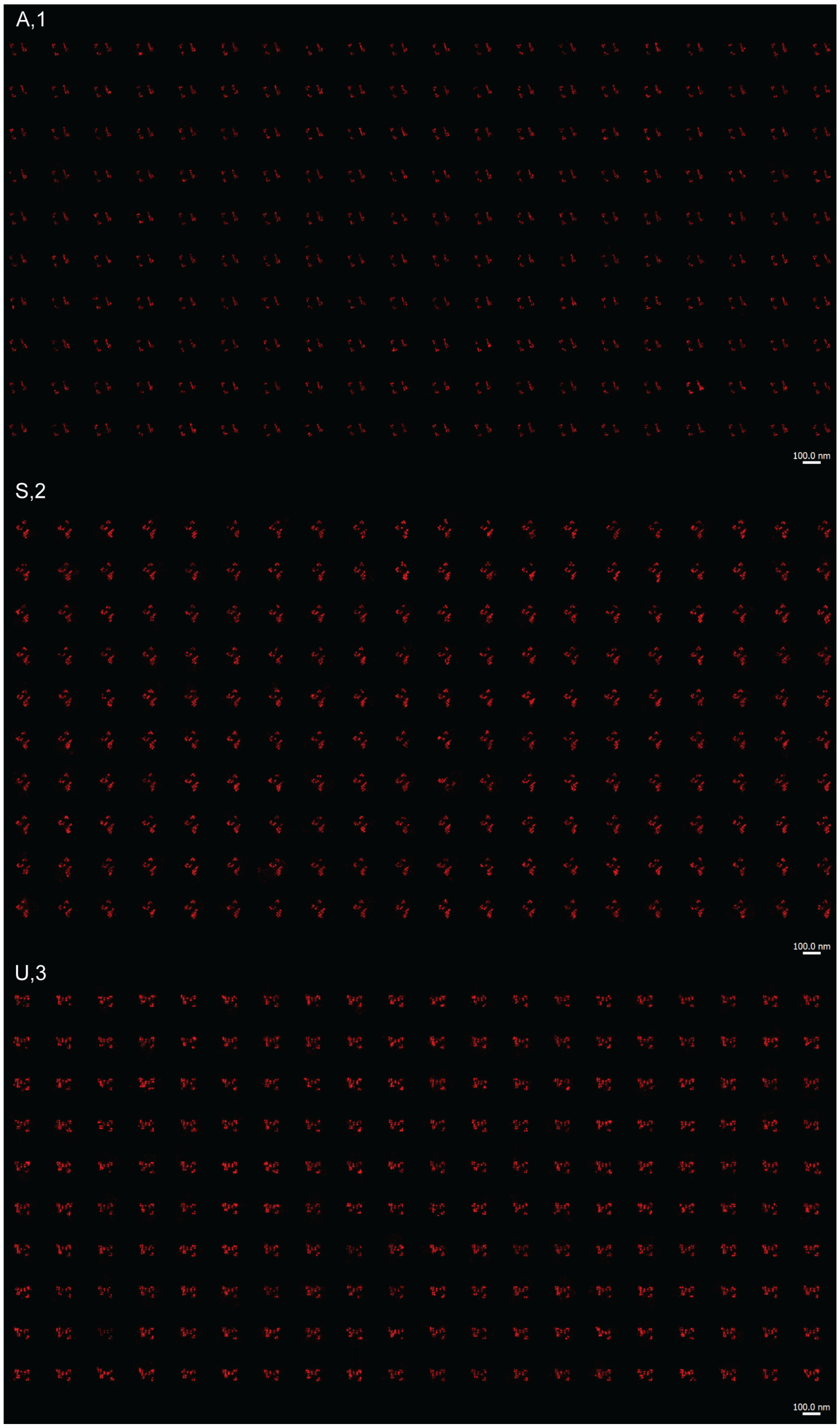


Fig. S6 **All picks in the analyzed "ASU" one redundancy dataset.** Full data set of 10 nm 1 redundancy encrypted "ASU" following Picasso Average alignment and Picasso render unfolding with a 100 nm scale bar.

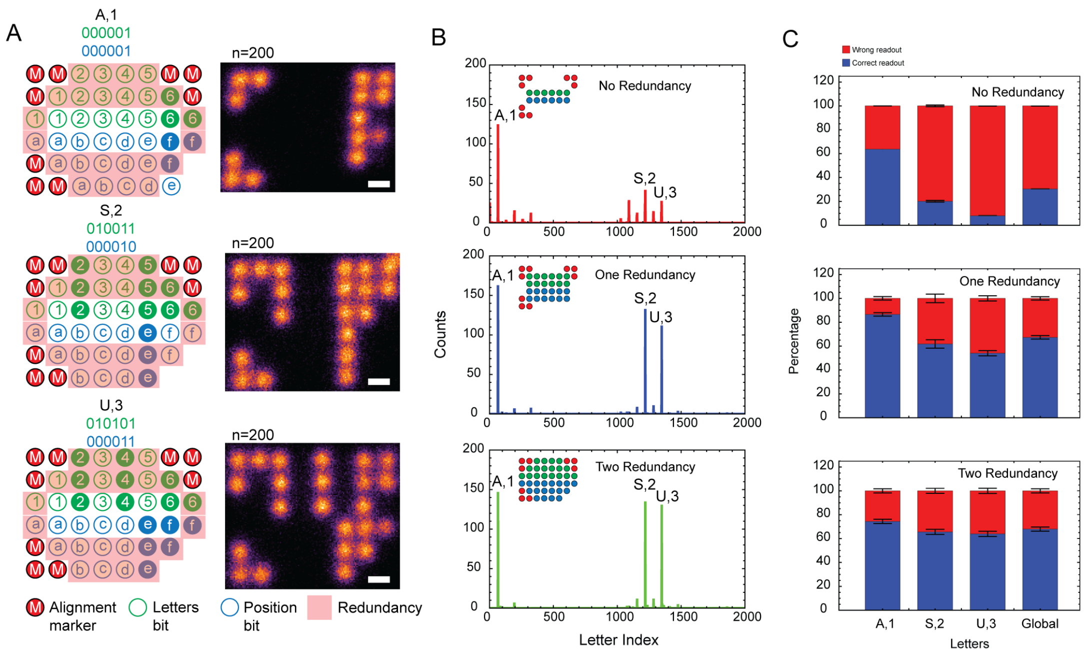


Fig. S7 **"ASU" two redundancy dataset on higher density 2D RRO along with All analyzed picks.** (**A**) The pattern encryption rules for "ASU" two redundancy dataset that shows the alignment marker, letters bit, position bit and the redundancy (left) and the summed DNA-PAINT images of three letters of "ASU" with the scale bar of 10 nm (right). (**B**) The readout of "ASU" dataset presented as letter index vs counts which are analyzed by not including the redundancy (top) and including the redundancy (middle and bottom). (**C**) The readout percentage of correct and wrong readout for each letter and global. The error bar is the standard deviation from three different processing runs with the same dataset.

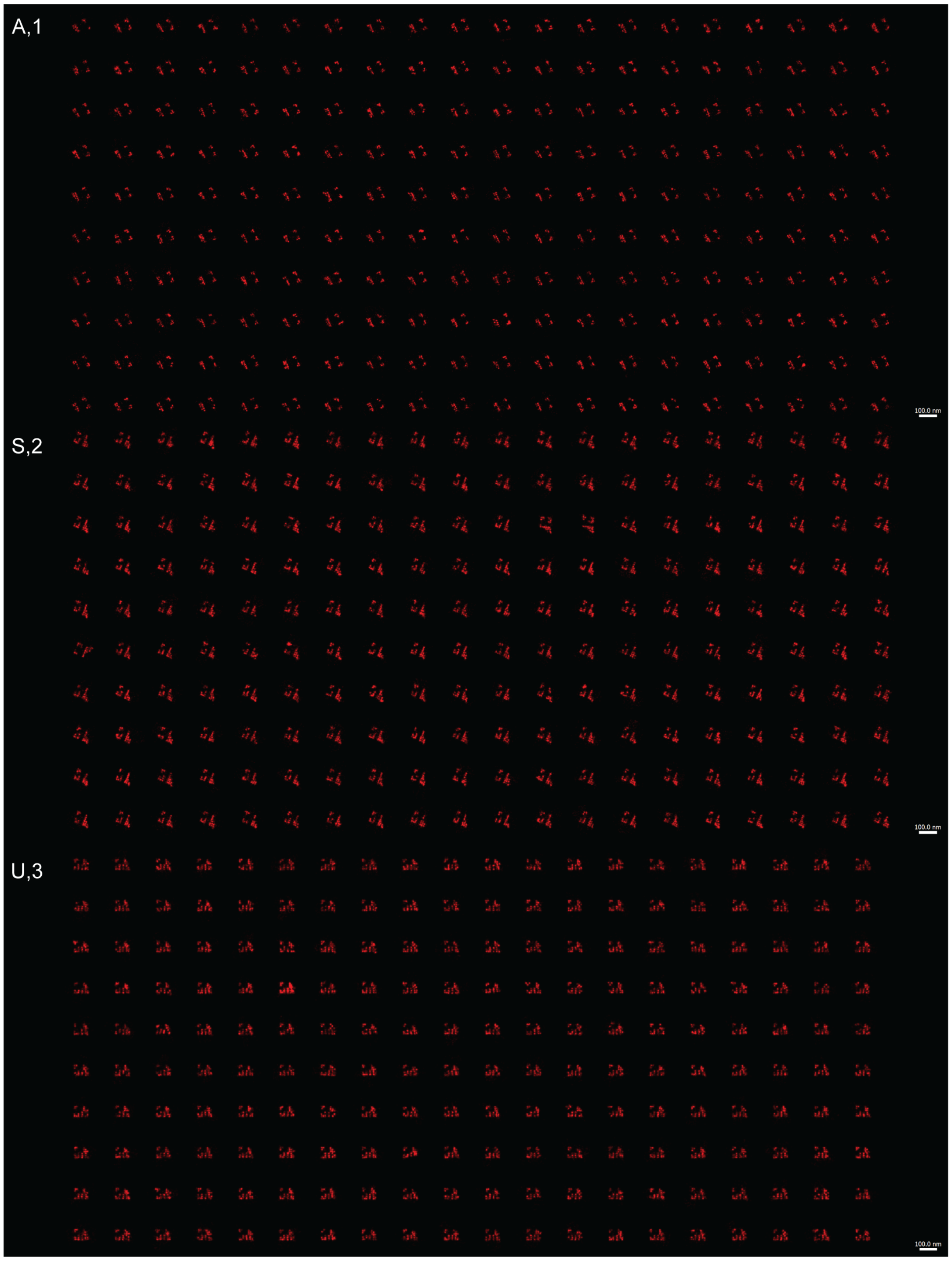
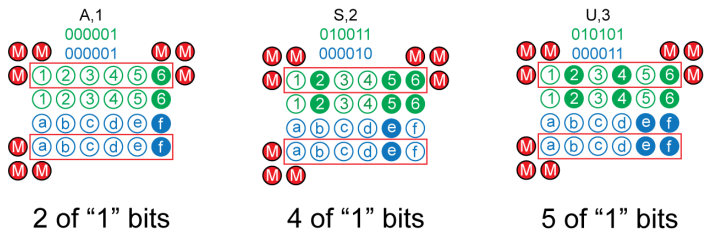


Fig. S8 **All analyzed picks in "ASU" two redundancy dataset on higher density 2D RRO.** Full data set of 10 nm 2 redundancy encrypted "ASU" following Picasso Average alignment and Picasso render unfolding with a 100 nm scale bar.



From Fig. 2, the detection efficiency of a docking is ~85-90%. Using 85%, the probability for a pair of dockings (in the case of one redundancy) to be at least one detected is $1-0.15^2=0.978$.

For A,1 pattern with 2 of "1" bits, the probability to have a correct pattern to be correctly read $0.978^2=0.956$. The same estimation goes for S,2 with 4 of "1" bits and U,3 with 5 of "1" bits that have probability of $0.978^4=0.915$ and $0.978^5=0.895$. However, there will be alignment accuracy, and K-means assignment accuracy as well that need to be considered. We can safely assume that alignment and K-means accuracy to be each around 90%. Therefore, both process will contribute to a ~ 81% accuracy. Using this value, the overall probability will be 0.956*0.81=0.77 for A,1 pattern, 0.915*0.81=0.74 for S,2 pattern and 0.895*0.81=0.72 for U,3 pattern. As we can see, the correct readout percentage decreases as we increase the number of bit "1". Although the values do not exactly match the experimental value which we think can be due to additional factor in alignment accuracy that tends to decrease as we have more "1" bits.

Fig. S9 **Argument of the readout accuracy with increasing bits usage by a pattern.** The encryption pattern schematic for "ASU" 1 redundancy with different bit "1" usage (top). The discussion on why the accuracy goes down as the bit usage increases.

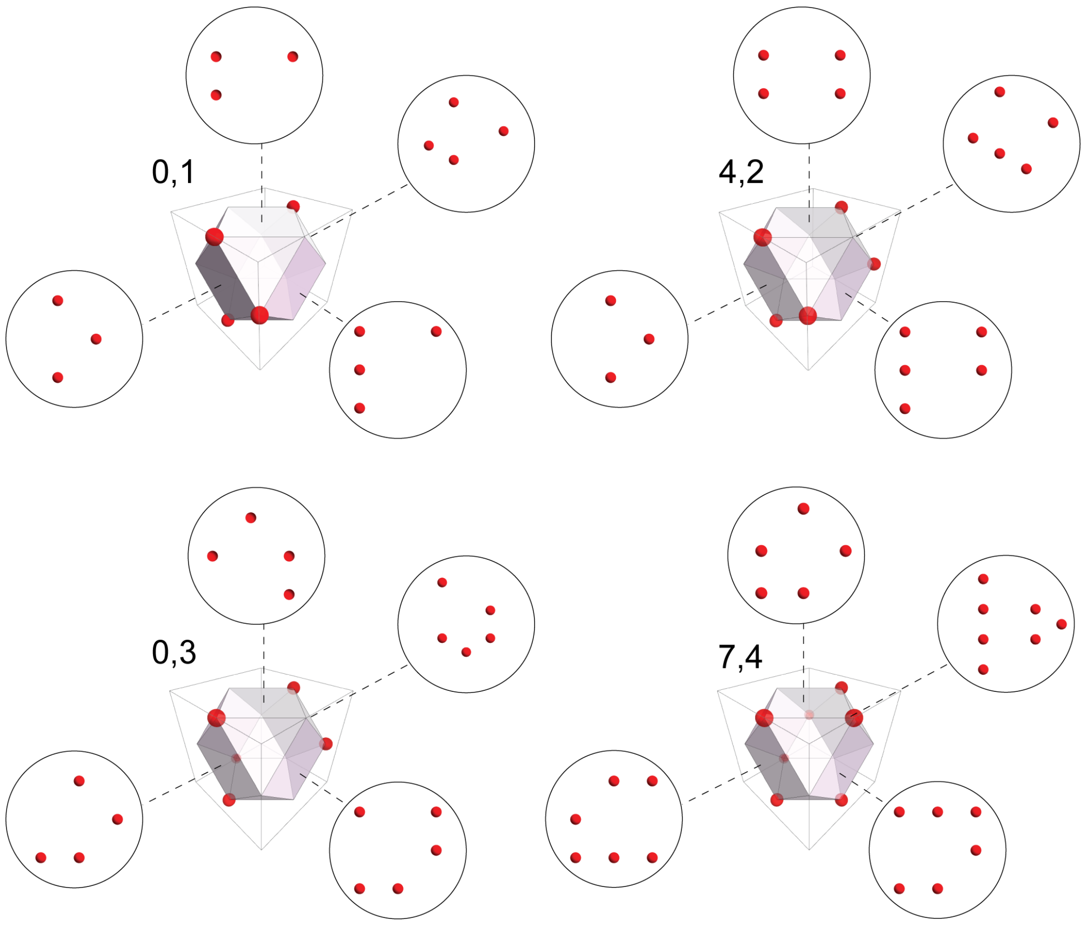


Fig. S10 Schematics of confused patterns due to 2D projections from 3D DNA origami encryption design. The biotinylated strands dictate the 2D projections of each pattern if only images using 2D DNA-PAINT.

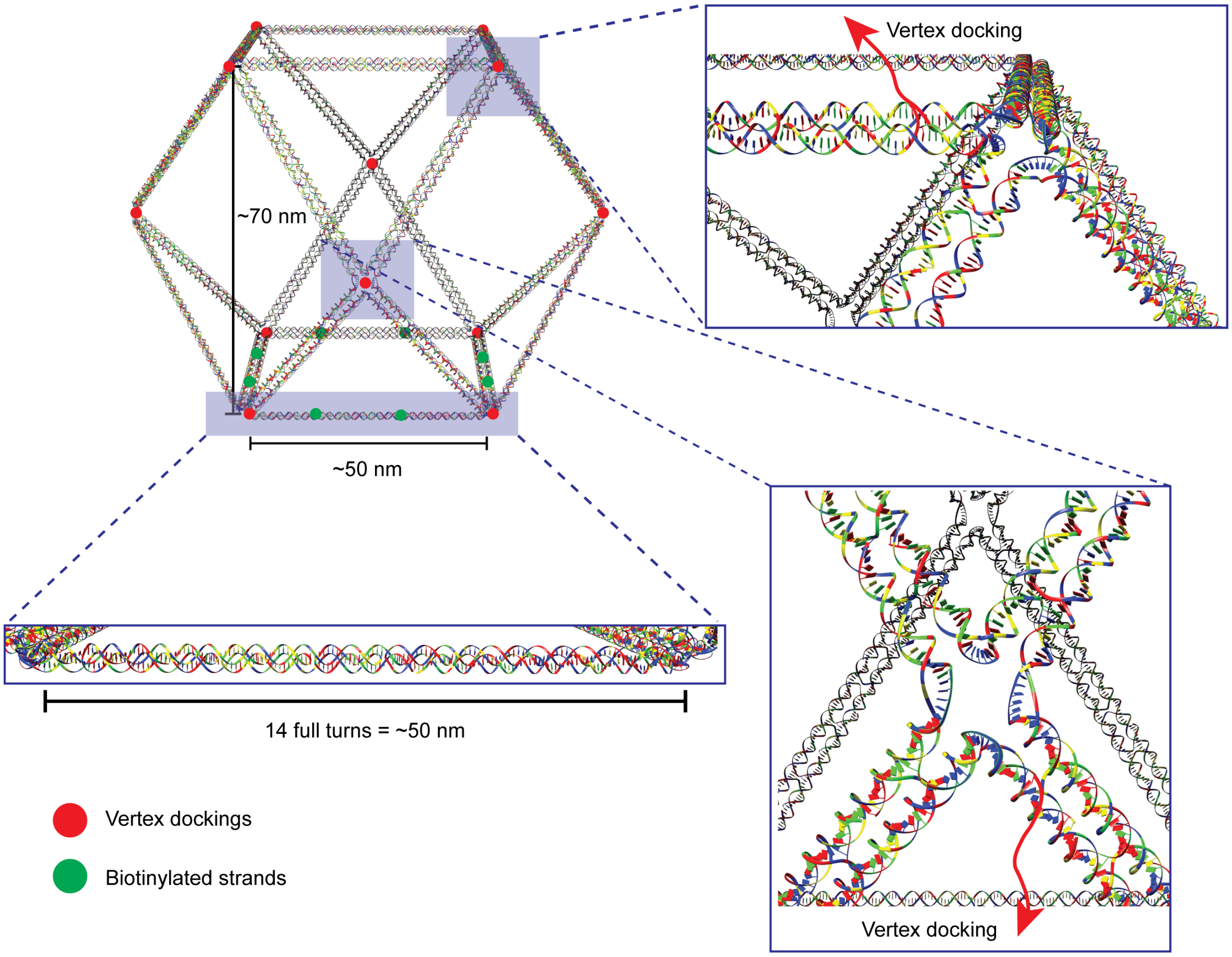


Fig. S11 **3D wireframe cuboctahedron DNA origami design.** The design of the 3D wireframe cuboctahedron with height of 70 nm and square faces with 50 nm side length due to 14 full turns of duplex (left). Vertex dockings and biotinylated strands are shown as red and green circles, respectively. Right panels show zoomed-in images of two vertices.

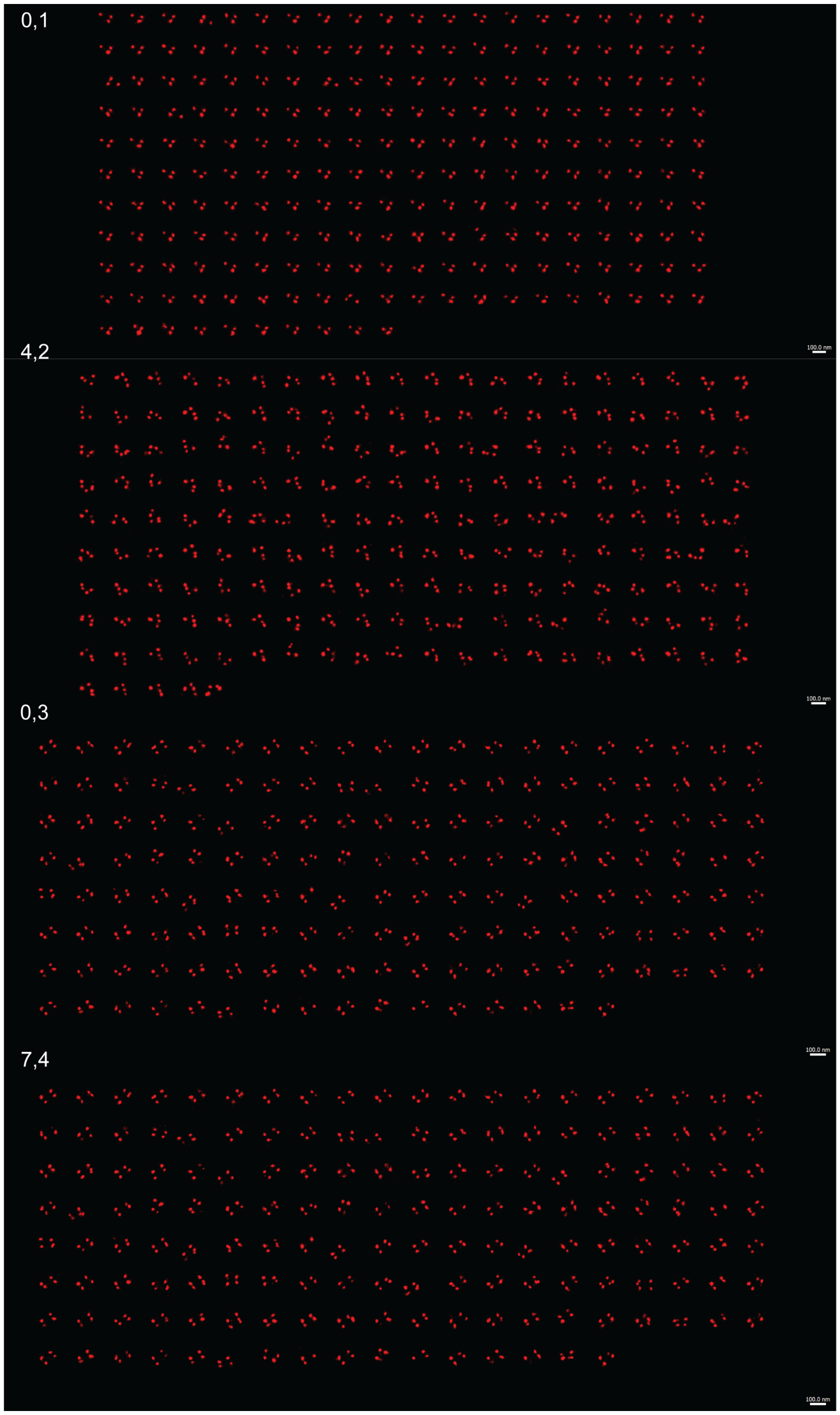


Fig. S12 **2D view of all picks analyzed in 3D DNA-PAINT "0407" dataset.** Full data set of 2D projection of "0407" dataset following Picasso Average alignment and Picasso render unfolding with a 100 nm scale bar.

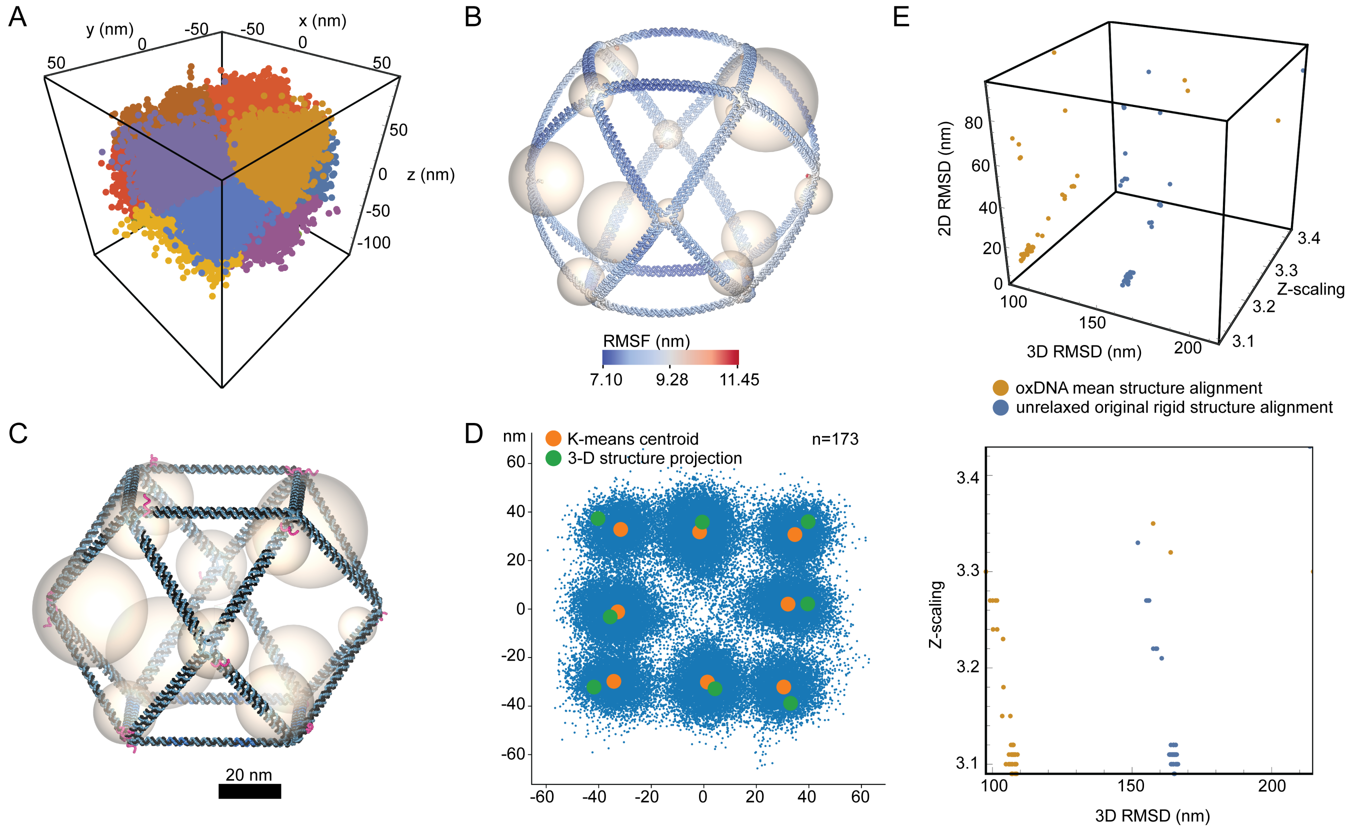
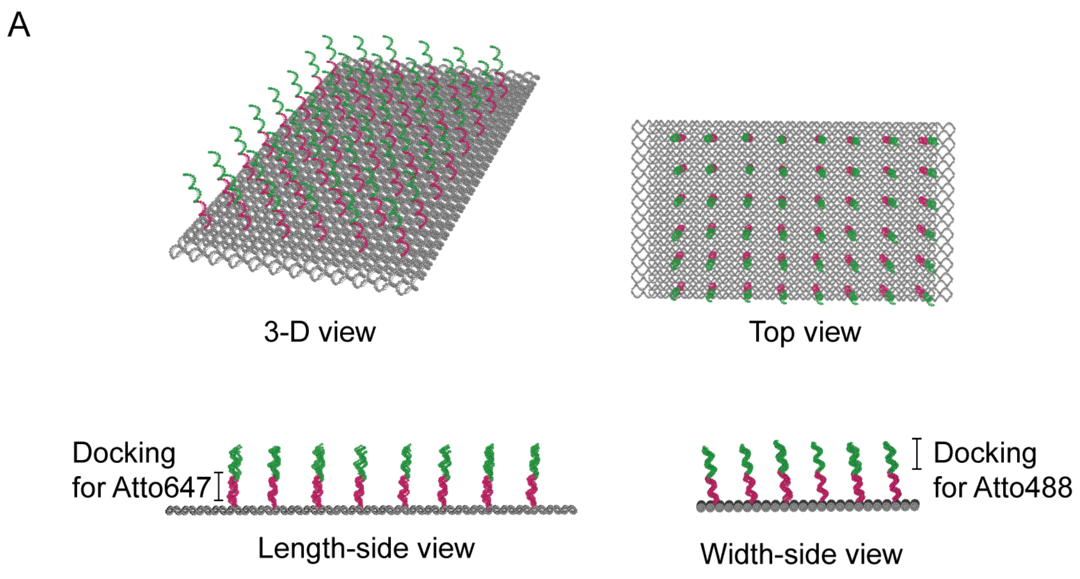
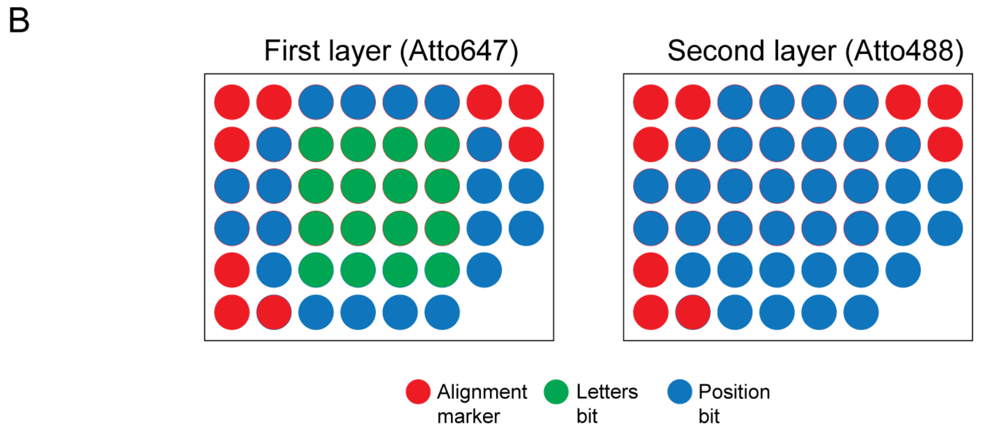


Fig. S13 **3D clustering and alignment of DNA-PAINT experimental data and 3D cuboctahedron DNA origami structure.** (**A**) 3D K-means clustering of 3D DNA-PAINT localization data by assigning K=12. (**B**) 3D Alignment of centroids from 3D K-means clustering result from **A** with the mean structure obtained from oxDNA simulation. The size of each sphere depicts the distance between the K-means centroid and the center of mass of the closest docking handle. (**C**) 3D Alignment of centroids from 3D K-means clustering result from **A** with the unrelaxed structure. The size of each sphere depicts the distance between the K-means centroid and the center of mass of the closest docking handle. **B** and **C** have the same scale with scalebar of 20 nm. (**D**) An example of 2D alignment of the unrelaxed structure with docking handles' center of mass projected to x-y plane (the two stacking docking handles in z direction are averaged). (**E**) Top panel: the plot of 3D RMSD vs 2D projection RMSD after 3D aligment vs the Z-scaling of mean and unrelaxed structure. Bottom panel: the plot of Z-scaling vs 3D RMSD showing the mean structure provides better RMSD thus better alignment with the K-means centroid of experimental DNA-PAINT data as compared to the unrelaxed structure alignment.





Docking: 48 first layer and 48 second layer

Alignment markers: 2x12 dockings

Characters bit: 2x8 dockings (8 bits encryption with one redundancy)

Position bit: 2x28 dockings (with one redundancy)

Fig. S14 **Design of two color 2D RRO encryption for high density information.** (**A**) The RRO schematic has two docking types for two different fluorophores. (**B**) The pattern encryption rules for two docking layers.

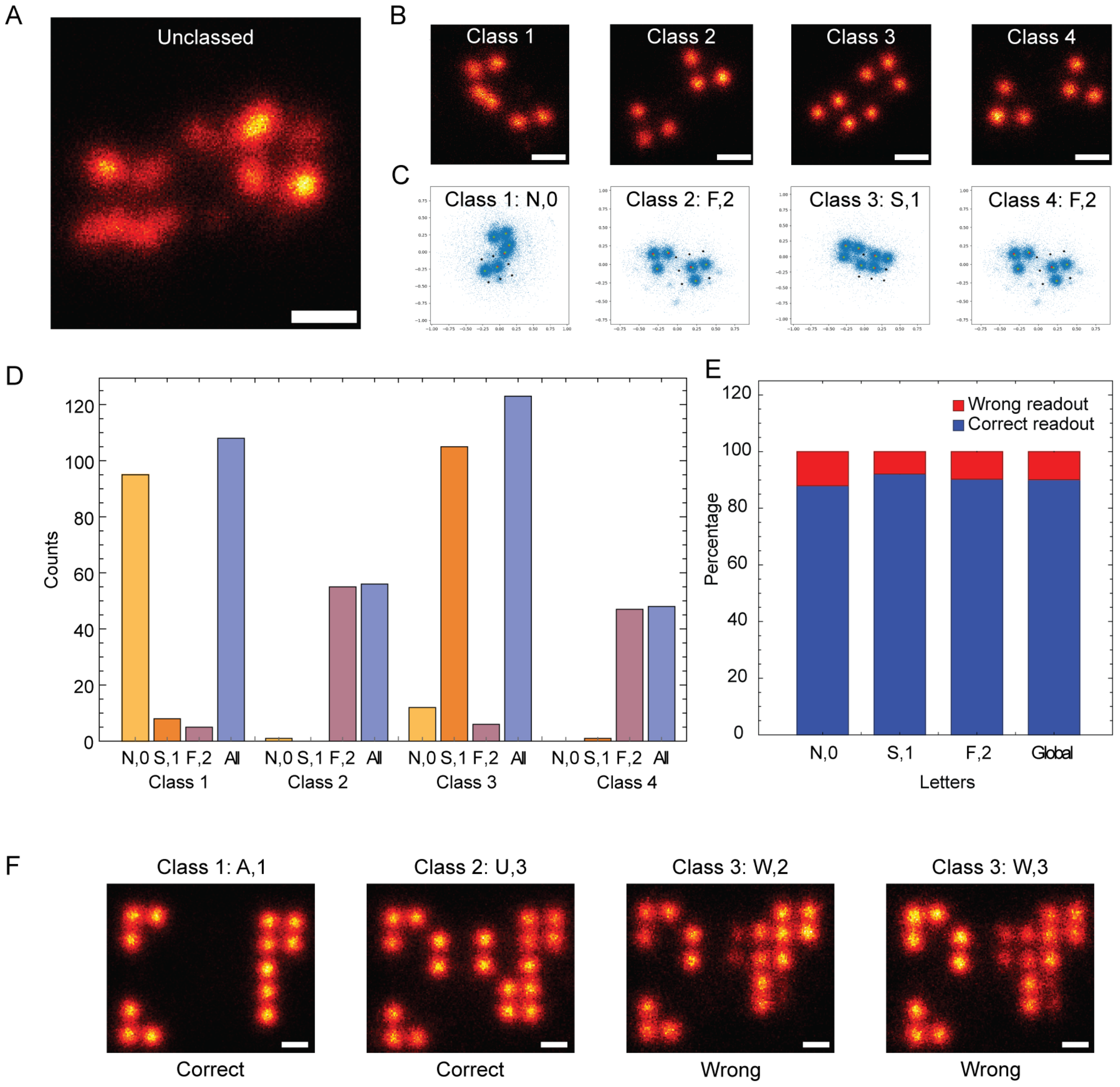
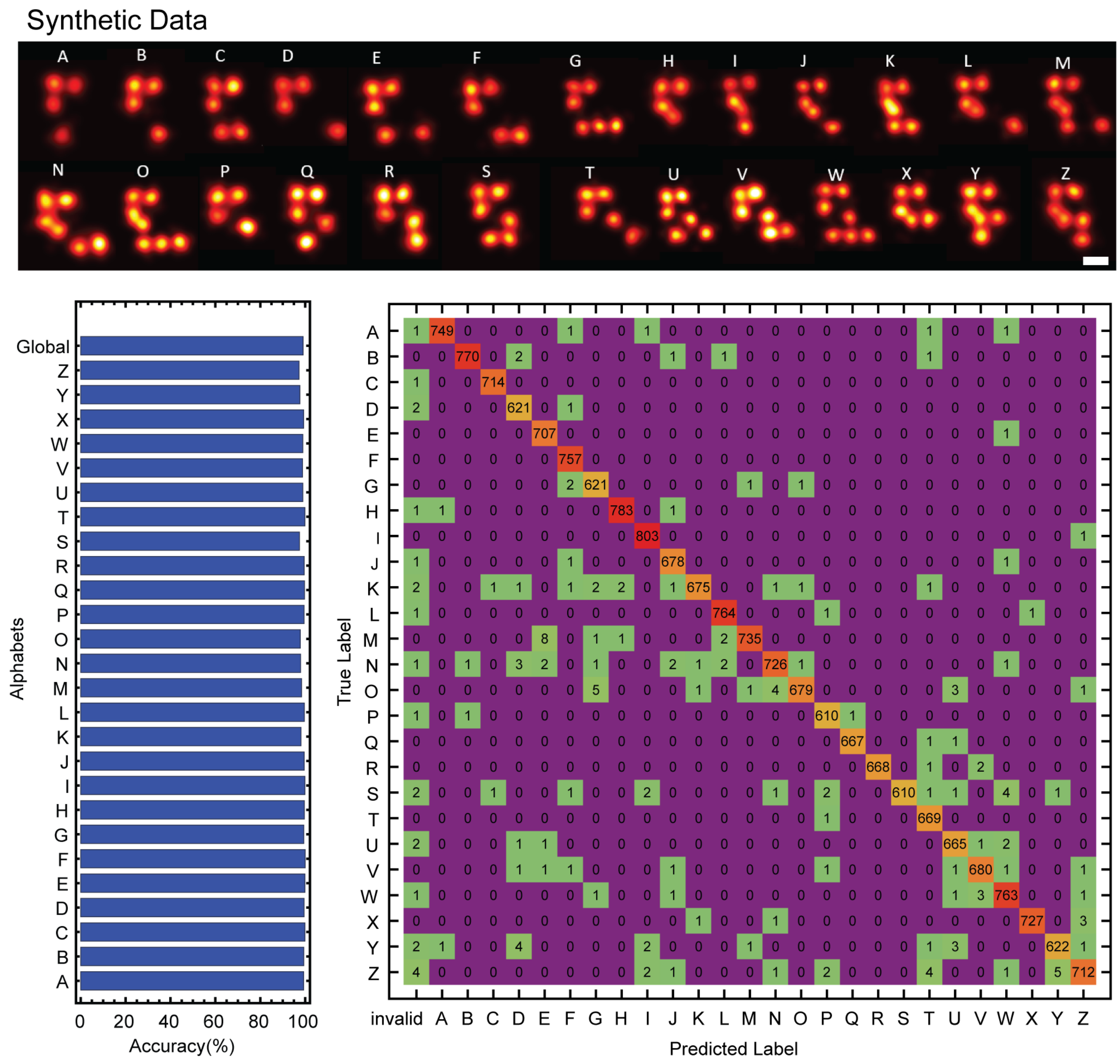


Fig. S15 **Unsupervised classification result using method described by Huijben et al. on "NSF" dataset.** (**A**) The superparticles of "NSF" before classification.(**B**) Superparticles of each class after classification showing 4 classes. (**C**) Readout of each class. (**D**) Classes' members showing a small number of miss-classed patterns thus not affecting the superparticles.(**E**) The readout accuracy of each class.(**F**) The classification of "ASU" one redundancy dataset showing two correct classes and two wrong classes thus unable to recover "ASU". Scale bar: 20 nm in (A) and (B), 10 nm in (F).



	invalid	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	1	749	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
B	0	0	770	0	2	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
C	1	0	0	714	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	2	0	0	0	621	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	707	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
F	0	0	0	0	0	0	757	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	621	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
H	1	1	0	0	0	0	0	0	783	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	803	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
J	1	0	0	0	0	0	1	0	0	0	678	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
K	2	0	0	1	1	0	1	2	2	0	1	675	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0
L	1	0	0	0	0	0	0	0	0	0	0	0	764	0	0	0	1	0	0	0	0	0	0	0	1	0	0
M	0	0	0	0	0	8	0	1	1	0	0	0	2	735	0	0	0	0	0	0	0	0	0	0	0	0	0
N	1	0	1	0	3	2	0	1	0	0	2	1	2	0	726	1	0	0	0	0	0	0	0	1	0	0	0
O	0	0	0	0	0	0	0	5	0	0	0	1	0	1	4	679	0	0	0	0	0	3	0	0	0	0	1
P	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	610	1	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	667	0	0	1	1	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	668	0	1	0	2	0	0	0	0
S	2	0	0	1	0	0	1	0	0	2	0	0	0	0	1	0	2	0	0	610	1	1	0	4	0	1	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	669	0	0	0	0	0	0
U	2	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	665	1	2	0	0	0
V	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	680	1	0	0	1
W	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	3	763	0	0	1
X	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	727	0	3
Y	2	1	0	0	4	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	1	3	0	0	0	622	1
Z	4	0	0	0	0	0	0	0	0	2	1	0	0	0	1	0	2	0	0	0	4	0	0	1	0	5	712

Fig. S16 **ResNet CNN results on synthetic data generated by Picasso Simulate module of letters A-Z without position encoding.** Examples of the 26 alphabets of synthetic data generated through Picasso Simulate module (top). The accuracy of each alphabets with ResNet-50 (bottom left) and the confusion matrix (bottom right).

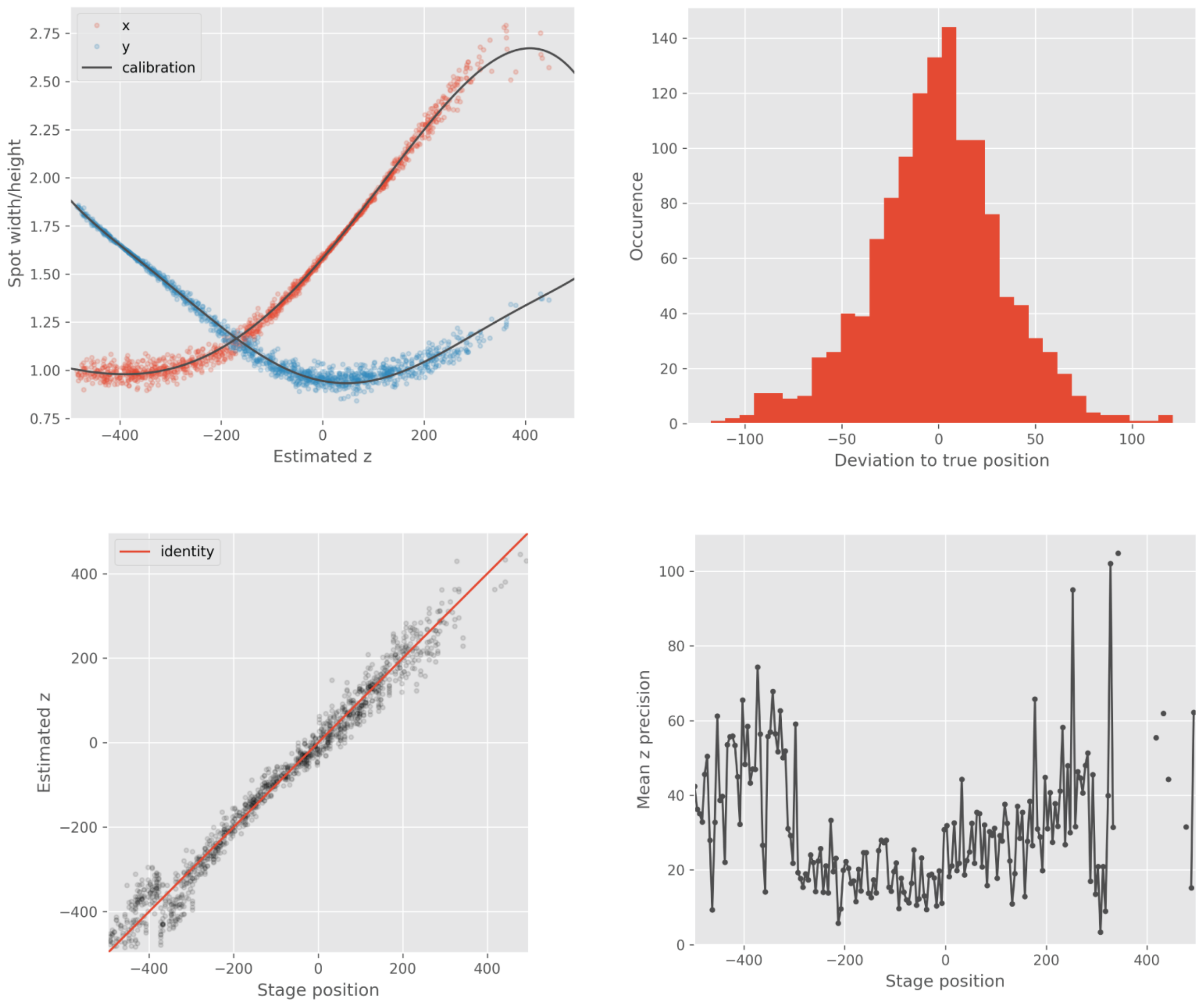


Fig. S17 **3D calibration curve generated by Picasso Localize.** The localizations spot widths and heights with the fit (top left). The distribution of the deviation to the true position (top right). The estimation of z coordinate as a function of stage position (bottom right). The mean z precision as a function of stage position (bottom right).