Digital Signatures to Ensure the Authenticity and Integrity of Synthetic DNA Molecules

Diptendu Mohan Kar
Dept. of Computer Science
Colorado State University
diptendu.kar@colostate.edu

Indrajit Ray
Dept. of Computer Science
Colorado State University and
U.S. National Science Foundation
indrajit.ray@colostate.edu

Jenna Gallegos
Chemical and Biological Engineering
Colorado State University
jenna.gallegos@colostate.edu

Jean Peccoud
Chemical and Biological Engineering
Colorado State University
jean.peccoud@colostate.edu

ABSTRACT

DNA synthesis has become increasingly common, and many synthetic DNA molecules are licensed intellectual property (IP). DNA samples are shared between academic labs, ordered from DNA synthesis companies and manipulated for a variety of different purposes, mostly to study their properties and improve upon them. However, it is not uncommon for a sample to change hands many times with very little accompanying information and no proof of origin. This poses significant challenges to the original inventor of a DNA molecule, trying to protect her IP rights. More importantly, following the anthrax attacks of 2001, there is an increased urgency to employ microbial forensic technologies to trace and track agent inventories. However, attribution of physical samples is next to impossible with existing technologies. In this paper, we describe our efforts to solve this problem by embedding digital signatures in DNA molecules synthesized in the laboratory. We encounter several challenges that we do not face in the digital world. These challenges arise primarily from the fact that changes to a physical DNA molecule can affect its properties, random mutations can accumulate in DNA samples over time, DNA sequencers can sequence (read) DNA erroneously and DNA sequencing is still relatively expensive (which means that laboratories would prefer not to read and re-read their DNA samples to get error-free sequences). We address these challenges and present a digital signature technology that can be applied to synthetic DNA molecules in living cells.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**; **Digital signatures**; **Hash functions and message authentication codes**; • **Applied computing** → *Molecular sequence analysis*; *Sequencing and genotyping technologies*;

KEYWORDS

Cyber-Bio Security, DNA, Identity Based Signatures, Reed-Solomon Codes

ACM Reference Format:
Diptendu Mohan Kar, Indrajit Ray, Jenna Gallegos, and Jean Peccoud. 2018. Digital Signatures to Ensure the Authenticity and Integrity of Synthetic DNA Molecules. In *New Security Paradigms Workshop (NSPW '18), August 28–31, 2018, Windsor, United Kingdom.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3285002.3285007

NSPW '18, August 28–31, 2018, Windsor, United Kingdom

ACM ISBN 978-1-4503-6597-0/18/08…$15.00
https://doi.org/10.1145/3285002.3285007

1 INTRODUCTION

DNA synthesis is becoming more commonplace, and samples containing DNA frequently change hands within the life sciences community. Samples are shared between academic labs, ordered from DNA synthesis companies, and manipulated for a variety of different purposes. It is becoming increasingly difficult to properly attribute DNA molecules to their original sources.

Example use cases where attribution of a DNA molecule to its originator is important are as follows: (i) Following the anthrax attacks of 2001, there has been an increased urgency to employ microbial forensic technologies to trace and track agent inventories. Many of these biological agents are created or manipulated in laboratories. (ii) Academic laboratories and biotech companies frequently treat synthetic DNA as licensed intellectual property that needs to be protected; the first step towards such protections is successfully ensuring attribution. (iii) In the future, if synthetic genes are used in gene therapy based medical treatment, such attribution could (a) readily inform the user about matters related to the therapy, and/or (b) serve as some measure of the quality of the therapy, à la brand name versus generic drugs. (iv) Recently, a DNA-based security exploit was demonstrated as a proof of concept, where synthetic DNA was used to attack a DNA sequencer that was deliberately modified with a vulnerability [8]; ensuring source attribution could help mitigate similar attacks on DNA sequencers.

The exact sequence of synthetic DNA molecules is generally documented electronically. Thus, one potential way (although indirect) of achieving such attribution is to include the origin information in the electronic document. However, the association between physical sequences and their electronic documentation is very loose [9, 11]. It is not uncommon for a sample to change hands many

times with only vague descriptions of where it originated or the exact sequences being shared. On the other hand, simply documenting samples more effectively can create security and intellectual property risks [12]. For instance, if the full sequence of pathogenic viral vectors was freely available, that information could be used with malicious intent. Or, for example, if a bio-tech company wants to keep some sequence information confidential to protect their IP rights, they may choose to exclude such information from the documentation. Regardless, the source of a given sample cannot be verified using only a careful documentation of the DNA sequence.

In the digital realm, the problem of authenticating a document or software while still withholding proprietary or sensitive information is achieved by digital signatures. Digital signatures are used to authenticate the source of a digital file and to confirm that the file has not been changed since the originator applied the signature. Our hypothesis is that digital signature technology can be used to solve the DNA attribution problem. Additionally, the same technology, can be used to detect mutations in DNA samples. Mutations are known to occur randomly at low frequencies, and they can compromise the function of DNA molecules. Recent works in DNA cryptography, such as, encoding arbitrary information in DNA [1], and embedding watermarks in synthetic DNA [3, 5–7, 15], gives us some degree of confidence in the validity of this hypothesis.

To solve the problem of tracing the source of synthesized DNA molecules and confirming their identity and integrity, we have developed a system for generating digital signatures for molecules of DNA in living cells. Specifically, we have used Shamir's Identity-based Signature (IBS) scheme [16]. For the unique identifier string of the originator, we use Open Researcher and Contributor ID's (ORCID) – https.orcid.org. ORCID is a non-profit organization which uniquely identifies researchers using a 16 digit number. Many funding agencies require researchers to register for an ORCID, and scholarly journals request that authors identify themselves using their ORCID. The generated signature bits are converted to the four letters A, C, G, and T, which represent the four nucleotide building-blocks of DNA. The sequence can then be synthesized and inserted into the original DNA molecule. When this signed molecule is shared, a receiver can sequence the signed molecule to verify that it was shared by an authentic sender and that the sequence of the original molecule has not been altered or tampered with.

While the use of these techniques in the digital world is quite common, applying them to DNA required several creative adjustments. The foremost challenge we faced arose from the physical size of the DNA sequence encoding the signature. Adding extraneous sequences to a DNA molecule can impact it's function or stability. It was, thus, important to minimize the size of the added sequence in order to decrease the likelihood that the biological function of the signed molecule would be effected and to decrease the cost of synthesizing the signature. This restricted our ability to use more well-known signature schemes as well as larger key sizes for signatures.

The second challenge was accounting for DNA mutations. In a DNA sample, mutations occur randomly at low frequencies, and, as a result, there is a non-trivial possibility that a signed molecule could undergo a mutation between the time it is signed and when it is validated. Mutations could affect not only the original DNA molecule but also the signature. In both cases, the signature validation will fail even if the molecule is sent by the correct authority and the original sequence was correct during the process of signature generation. Mutations are beyond the control of any authority and the relative impact of any given mutation can vary. In order to address this concern, we included error correction codes to detect mutations in the signed DNA molecule. Error correction codes are prevalent in digital storage such as CD/DVD. It is possible to use the same techniques to provide a reliable reconstruction of the original sequence for comparison, provided a small number of changes have occurred. Our application of error correction codes to DNA could also be used to ensure the integrity of digital information stored in DNA molecules.

Finally, while digital signatures provide a way to verify the source and integrity of a DNA sample, there is additional information about the DNA sequence that would be very useful to the recipient of a signed DNA sample. For example, the exact location of features within the sequence, such as a certain gene, will still be unknown to the recipient. We thus developed a method to link the physically signed DNA molecule with its digital representation, which contains the sequence and its features with explanations.

Threat Model: For this work, we assume a polynomial-time adversary, Mallory, who is trying to forge the signature of a reputed synthesized DNA molecule creator, Alice. Alice is trying to protect her IP rights / reputation as she distributes DNA molecules synthesized by her to researcher Bob. If the attacker, Mallory, is able to forge the signature of Alice then: (a) Mallory can replace the actual DNA created by Alice with her own but keep the signature intact. (b) Mallory can create her own DNA molecule and masquerade as Alice to sign it. (c) Mallory can modify parts of the signed DNA molecule created by Alice. Therefore, for our proposed signature scheme to be secure, we need to show that no polynomial-time adversary can forge a genuine signature without knowing the secret used to sign.

The rest of the paper is organized as follows. In Section 2, we discuss relevant works in DNA cryptography, especially those that use watermarking techniques for DNA origin authentication. Section 3 gives a small overview of the biological domain which forms the backdrop of our scheme. Section 4 discusses the challenges we faced and our design decisions. In Section 5, we present our vanilla signature scheme and describe how the digital signature is embedded in the physical DNA molecule. Section 6 describes the workflow that we have to follow in order to validate the signed physical DNA molecule. Section 7 discusses how error correction code is used to address the problem of mutations. In Section 9, we describe the process of strongly tying the digital document corresponding to a signed DNA molecule with the physical DNA sample. We include some discussions on the practicality and usability of our signature scheme in Section 10. Finally, we conclude in Section 11 by briefly talking about validation of our scheme and discussing some open problems that we can pursue in this area.

2 RELATED WORK

By converting the binary system of 0's and 1's used by computers into the four-letter genetic code (A-C-G-T), encryption algorithms

have been developed to store the content of a book, an operating system, and even a movie in synthetic DNA [1, 2]. Computer scientists working in the field of DNA computing have also proposed developing asymmetric encryption schemes using DNA sequences [17]. These pioneering projects demonstrate that DNA synthesis and DNA sequencing technologies are able to translate between DNA molecules and digital information. Our proposed approach is the first application of a digital signature scheme to DNA in a living organism, which will provide a physical significance to a digital signature.

There is a growing body of research concerned with traceability and unique identification of the source of DNA sequences. For instance, it has been proposed that unique watermarks be inserted in the genome of infectious agents to increase their traceability [6]. The synthetic genomics community has demonstrated the feasibility of this approach by inserting short watermarks into DNA without introducing significant perturbation to genome function [3, 5, 7, 15]. The use of watermarks has also been proposed in order to identify genetically modified organisms (GMOs) or proprietary strains. Our digital signature-based approach provides a much greater level of security than watermarking because a watermark is independent of the sequence it is attached to (only changes to the watermark itself would be detectable) and watermarks are easily counterfeited.

Heider et al. [4] describes DNA-based watermarks using DNA-Crypt algorithm. This technique is applicable to provide proof of origin to a DNA molecule. However, there are some limitations. First, the watermark is generated from any binary data and added to the original sequence. The watermark is independent of the original sequence and hence provides no integrity to the actual DNA sequence. The sequence of the original molecule could change (intentionally or due to mutations) while the watermark remains unaltered. Anyone attempting to validate the watermark would not be able to tell that the sequence had changed. Second, if an attacker knows the binary data that was used to generate the watermark, they could generate their own DNA sequence and include the watermark to malign the actual user/organization. An attacker could also apply their own watermark to proprietary DNA sequences that are licensed by another entity. Owing to these reasons, we cannot just rely on watermarks to ensure authenticity. Instead, we use digital signatures as they provide much better and stronger security guarantees. Finally, DNA-Crypt uses some symmetric key encryptions like AES, Blowfish to encrypt the binary data that is used to create the watermark. Hence, these keys have to be transmitted to the receiver who will validate the watermark. DNA-Crypt has options to export and import keys and exchange them with receivers. But the receiver now has the secret key that was used to generate the watermark and can masquerade as the originator of the DNA. Non-repudiation becomes a problem too. Although DNA-crypt mentions that RSA algorithm can be used as an asymmetric scheme, the developers do not talk about the challenges involved when using RSA since the length of the watermarks will be dependent on the RSA security parameter.

3 BACKGROUND

As a first step, we have applied digital signatures to plasmids. A plasmid is a small DNA molecule within a cell that is physically separated from the chromosomal DNA and can replicate independently. Plasmids are small, circular, double-stranded DNA molecules commonly found in bacteria. In nature, plasmids often carry genes that may benefit the survival of the organism, for example, genes for antibiotic resistance. While chromosomes are big and contain all the essential genetic information for living under normal conditions, plasmids usually are very small and contain only additional genes that may be useful to the organism under certain situations or particular conditions. Artificial plasmids are widely used as vectors in molecular cloning, serving to drive the replication of recombinant DNA sequences within host organisms. The reason for starting with plasmids are -

- Plasmids can be isolated in large quantities.
- We can cut and splice them, adding whatever DNA we choose.
- We can put them back into bacteria, where they'll replicate along with the bacteria's own DNA.
- We can isolate them again - getting billions of copies of whatever DNA we inserted into the plasmid!

Plasmids are generally limited to sizes of 2.5-20 kilobases (each letter of the genetic code A-C-G-T is 1 base).

The sequences that make up a plasmid are often documented electronically. Molecular biologists are now equipped with automated DNA sequencers that identify the pattern of bases in a physical DNA sample and document the sequence in a digital file called a fasta file (`.fasta`). A sample fasta file is shown in Figure 1. The sequences can then be converted to annotated files such as `.dna`, or `.gb` files that include information about what genetic features are included in the plasmid. A sample genbank (`.gb`) file is shown in Figure 2. Each of these files has a specific format which denotes the sequences together with other pieces of information including the location of features such as coding sequences (CDS), origin of replication, etc. Sequence-manipulation softwares such as SnapGene (http://www.snapgene.com) convert sequences into maps of plasmid features. Figure 3 depicts a map generated in SnapGene of the same digital DNA file depicted in Figure 2. Features (shown as colored block arrows in Figure 3) can be added manually or identified automatically by searching within the SnapGene database for common features. Not all of the sequences contain features. There are substrings or subsequences that do not have any known biological function. Biologists can add other DNA sequences in these areas with reasonable confidence that the added sequences will not disrupt the activity of any existing features.

Although the sequences that make up a plasmid can be documented electronically, the electronic sequence file associated with a physical DNA sample is seldom shared along with the sample it represents [10]. In life sciences manuscripts, plasmids are generally described one of four ways. Most often, the main features of the plasmid relevant to the publication are broadly explained (ie. "A plasmid containing gene X was used..."). Sometimes there is a more thorough description of how the plasmid was constructed included in the methods section (ie. "Gene X was inserted into a commercial plasmid between Origin Y and antibiotic resistance gene Z"). Full

plasmid maps (as shown in Figure 3) are very rarely included in published manuscripts, and inclusion of the full sequence – which you would need to validate the plasmid – is even more rare. Additionally, it is not uncommon for a plasmid to be shared multiple times between many labs until it is no longer clear who made the original plasmid. Even within a lab, it is often difficult to track down the digital sequence file associated with a plasmid if the person who constructed it is no longer an active member. The ability to validate a physical DNA sample without having access to the digital sequence file associated with it would, thus, be extremely valuable to the life sciences community.

We propose digital signatures as a strategy for encoding the ability to validate a physical DNA sample within the DNA itself. Once the sequences are in a digital file, we can apply digital signatures on the extracted sequence (message). The signature bits are then converted to ACGT sequence as A-00, C-01, G-10, and T-11 and added to the original sequence.

Once the signed sequence is obtained by adding the signature to the original sequence, it can be outsourced to a gene synthesis company that will synthesize the signed DNA and send it. The signature alone can be synthesized and inserted into the original molecule, or the entire plasmid can be synthesized including the signature to eliminate the need for any downstream assembly.

Digital DNA file formats

An automated DNA sequencer provides the digital representation of the sequences present within the physical sample. The output of a DNA sequencer is a fasta (`.fasta`) file as shown in Figure 1. This file contains only the raw sequences in the sample. A genbank file (`.gb`) contains the same raw sequences along with annotations. In Figure 2, after the word "ORIGIN" the raw sequences are denoted and before that the features are annotated.

Sequence manipulation software such as SnapGene can be used to convert a fasta file to a genbank file and vice versa. When a fasta file is converted to a genbank file, the software searches its database for common annotations. The generated annotations may not be complete or correct every time. Hence, the user has the flexibility to manually add additional annotations that may be required to describe the sample sequence. These manually added annotations are only available to the creator. When the same sample is sent to others, they will sequence it and obtain the fasta file but the genbank file will contain only those annotations that can be automatically generated. In order for the receiver to extract all the feature information for a given plasmid, the creator would need to share the genbank file containing the manually added annotations. In this work, we assume that the sender of the DNA sample will also share the genbank file, although this is very often not the case. (We are looking into ways of eliminating this step such that the genbank file can be generated from the fasta file itself.)

Circular DNA and reverse complement

Plasmid DNA is circular and double-stranded. The sequences represented in a fasta file are the linear representation of a circular structure. As a consequence, there is no single set representation of the sequences in a sample. Following sequencing, any cyclic permutation of the sequence is possible. For example, in a fasta file if the sequence is - "ACGGTAA", when the same sample is sequenced again, the fasta file might read as - "TAAACGG".

Furthermore, since DNA is composed of two complimentary, anti-parallel strands, a sequencer can read a sample in both the "sense" or "antisense" direction. The sequence may be represented in a fasta file in either direction. When the sample is sequenced again, the output might be in the other direction, or what is known as the reverse complement. The reverse complement of "A" is "T" and vice-versa, and the reverse complement of "C" is "G" and vice-versa. The DNA molecule has polarity with one end represented as 5' and the other represented as 3'. One strand adheres to its reverse compliment in anti-parallel fashion. So if the sequence is - "5'-ACGGTAA-3"', the reverse complement is "3'-TGCCATT-5"'. The fasta file will represent one strand of the DNA sequence in the 5' to 3' direction; so the fasta file could read as "ACGGTAA" or "TTACCGT". By combining these two properties, for a DNA that contains N number of bases, the correct representation of the same sample is $2N$: N cyclic permutations plus each reverse complement.

4 CHALLENGES AND DESIGN DECISIONS

In the digital realm, the application of digital signatures, nowadays, is trivial. But embedding a signature within a physical DNA molecule is challenging for a number of reasons. We describe these challenges and how we handled them below.

4.1 Signature length

For any digital asset, e.g. a digital document, the length of the signature does not affect the asset that is being signed. When applying digital signatures to DNA, we cannot use any arbitrary length for the signature. Inserting any extraneous DNA sequences could impact the function or stability of the DNA molecule. For instance, the inserted sequences could: 1) disrupt existing functions by interrupting important features, 2) introduce a new function by encoding cryptic functional elements or 3) impact the overall stability of the plasmid in terms of propensity for mutations, structural rearrangements or retention in the host organism. The probability that existing features will be disrupted can be minimized through careful choice of where within the plasmid the sequence is inserted. The probability that the inserted sequence could introduce a new function or impact stability increases with the length of the inserted sequence. Additionally, the cost of synthesizing the signature increases with length. However, it is not preferable to use weak security parameters to shorten the signature length as this might keep the properties of the molecule intact but instead compromise the security of the signature itself. For a digital document, a signature of 384 bytes (say) is trivial. But the same 384 bytes translates to 1536 bases (384 * 8 / 2) of DNA. If a DNA sample originally contains, say, 2000 bases (not unusual for a plasmid), the addition of a 1536 nucleotide signature would nearly double the size of the DNA molecule. As a consequence, we did not apply identity-based signatures that use bilinear pairings in order to minimize the size of the insertion. Hence, we decided to utilize Shamir's IBS scheme with a minor modification. The details of our signature scheme are described in section 5.

```
>pUC19.gb(2686bp)
tcgcgcgtttcggtgatgacggtgaaaacctctgacacatgcagctcccggagacggtcacagcttgtctgtaagcggatgccgggagcagacaagcccgtcagggcgcgtcag
cgggtgttggcgggtgtcggggctggcttaactatgcggcatcagagcagattgtactgagagtgcaccatatgcggtgtgaaataccgcacagatgcgtaaggagaaaataacc
gcatcaggcgccattcgccattcaggctgcgcaactgttgggaagggcgatcggtgcgggcctcttcgctattacgccagctggcgaaaggggatgtgctgcaaggcgattaa
gttgggtaacgccagggttttcccagtcacgacgttgtaaaacgacggccagtgaattcgagctcggtacccggggatcctctagagtcgacctgcaggcatgcaagcttggcg
taatcatggtcatagctgtttcctgtgtgaaattgttatccgctcacaattccacacaacatacgagccggaagcataaagtgtaaagcctggggtgcctaatgagtgagctaa
ctcacattaattgcgttgcgctcactgcccgctttccagtcgggaaacctgtcgtgccagctgcattaatgaatcggccaacgcgcggggagaggcggtttgcgtattgggcgc
tcttccgcttcctcgctcactgactcgctgcgctcggtcgttcggctgcggcgagcggtatcagctcactcaaaggcggtaatacggttatccacagaatcaggggataacgca
ggaaagaacatgtgagcaaaaggccagcaaaaggccaggaaccgtaaaaaggccgcgttgctggcgtttttccataggctccgccccctgacgagcatcacaaaaatcgacgc
tcaagtcagaggtggcgaaacccgacaggactataaagataccaggcgtttcccccctggaagctccctcgtgcgctctcctgttccgaccctgccgcttaccggatacctgtcc
gcctttctcccttcgggaagcgtggcgctttctcatagctcacgctgtaggtatctcagttcggtgtaggtcgttcgctccaagctgggctgtgtgcacgaaccccccgttcag
cccgaccgctgcgccttatccggtaactatcgtcttgagtccaacccggtaagacacgacttatcgccactggcagcagccactggtaacaggattagcagagcgaggtatgta
ggcggtgctacagagttcttgaagtggtggcctaactacggctacactagaagaacagtatttggtatctgcgctctgctgaagccagttaccttcggaaaaagagttggtagc
tcttgatccggcaaacaaaccaccgctggtagcggtggttttttttgtttgcaagcagcagattacgcgcagaaaaaaaggatctcaagaagatcctttgatcttttctacgggg
tctgacgctcagtggaacgaaaactcacgttaagggattttggtcatgagattatcaaaaggatcttcacctagatccttttaaattaaaaatgaagttttaaatcaatctaa
agtatatatgagtaaacttggtctgacagttaccaatgcttaatcagtgaggcacctatctcagcgatctgtctatttcgttcatccatagttgcctgactccccgtcgtgtag
ataactacgatacgggagggcttaccatctggccccagtgctgcaatgataccgcgagacccacgctcaccggctccagatttatcagcaataaaccagccagccggaagggcc
gagcgcagaagtggtcctgcaactttatccgcctccatccagtctattaattgttgccgggaagctagagtaagtagttcgccagttaatagtttgcgcaacgttgttgccatt
gctacaggcatcgtggtgtcacgctcgtcgtttggtatggcttcattcagctccggttcccaacgatcaaggcgagttacatgatcccccatgttgtgcaaaaaagcggttagc
tccttcggtcctccgatcgttgtcagaagtaagttggccgcagtgttatcactcatggttatggcagcactgcataattctcttactgtcatgccatccgtaagatgcttttct
gtgactggtgagtactcaaccaagtcattctgagaatagtgtatgcggcgaccgagttgctcttgcccggcgtcaatacgggataataccgcgccacatagcagaactttaaaa
gtgctcatcattggaaaacgttcttcggggcgaaaactctcaaggatcttaccgctgttgagatccagttcgatgtaacccactcgtgcacccaactgatcttcagcatcttttt
actttcaccagcgtttctgggtgagcaaaaacaggaaggcaaaatgccgcaaaaagggaataagggcgacacggaaatgttgaatactcatactcttcctttttcaatattat
tgaagcatttatcagggttattgtctcatgagcggatacatatttgaatgtatttagaaaaataaacaaatagggggttccgcgcacatttccccgaaaagtgccacctgacgtc
taagaaaccattattatcatgacattaacctataaaaataggcgtatcacgaggccctttcgtc
```

Figure 1: Sample fasta (.fasta) file

```
LOCUS       Exported                2686 bp ds-DNA     circular SYN 24-NOV-2013
DEFINITION  Standard E. coli vector with a multiple cloning site (MCS) for DNA
            cloning. The MCS is reversed in pUC18.
ACCESSION   .
VERSION     .
KEYWORDS    pUC19
SOURCE      synthetic DNA construct
  ORGANISM  synthetic DNA construct
REFERENCE   1  (bases 1 to 2686)
  AUTHORS   Yanisch-Perron C, Vieira J, Messing J.
  TITLE     Improved M13 phage cloning vectors and host strains: nucleotide
            sequences of the M13mp18 and pUC19 vectors.
  JOURNAL   Gene 1985;33:103-19.
  PUBMED    2985470
REFERENCE   2  (bases 1 to 2686)
  AUTHORS   New England Biolabs
  TITLE     Direct Submission
  JOURNAL   Exported Nov 10, 2017 from SnapGene 4.1.0
            http://www.snapgene.com
COMMENT     See also GenBank accession L09137.
FEATURES             Location/Qualifiers
     source          1..2686
                     /organism="synthetic DNA construct"
                     /lab_host="Escherichia coli"
                     /mol_type="other DNA"
     CDS             complement(146..469)
                     /codon_start=1
                     /gene="lacZ"
                     /product="LacZ-alpha fragment of beta-galactosidase"
                     /label=lacZ-alpha
                     /translation="MTMITPSLHACRSTLEDPRVPSSNSLAVVLQRRDWENPGVTQLNR
                     LAAHPPFASWRNSEEARTDRPSQQLRSLNGEWRLMRYFLLTHLCGISHRIWCTLSTICS
                     DAA"
     primer_bind     379..395
                     /label=M13 fwd
                     /note="common sequencing primer, one of multiple similar
                     variants"
     misc_feature    396..452
                     /label=MCS
                     /note="pUC19 multiple cloning site"
     primer_bind     complement(465..481)
                     /label=M13 rev
                     /note="common sequencing primer, one of multiple similar
                     variants"
     promoter        complement(2487..2591)
                     /gene="bla"
                     /label=AmpR promoter
ORIGIN
        1 tcgcgcgttt cggtgatgac ggtgaaaacc tctgacacat gcagctcccg gagacggtca
       61 cagcttgtct gtaagcggat gccgggagca gacaagcccg tcagggcgcg tcagcgggtg
      121 ttggcgggtg tcggggctgg cttaactatg cggcatcaga gcagattgta ctgagagtgc
      181 accatatgcg gtgtgaaata ccgcacagat gcgtaaggag aaaataccgc atcaggcgcc
      241 attcgccatt caggctgcgc aactgttggg aagggcgatc ggtgcgggcc tcttcgctat
      301 tacgccagct ggcgaaaggg ggatgtgctg caaggcgatt aagttgggta acgccagggt
      361 tttcccagtc acgacgttgt aaaacgacgg ccagtgaatt cgagctcggt acccggggat
      421 cctctagagt cgacctgcag gcatgcaagc ttggcgtaat catggtcata gctgtttcct
      481 gtgtgaaatt gttatccgct cacaattcca cacaacatac gagccggaag cataaagtgt
      541 aaagcctggg gtgcctaatg agtgagctaa ctcacattaa ttgcgttgcg ctcactgccc
      601 gctttccagt cgggaaacct gtcgtgccag ctgcattaat gaatcggcca acgcgcgggg
      661 agaggcggtt tgcgtattgg gcgctcttcc gcttcctcgc tcactgactc gctgcgctcg
      721 gtcgttcggc tgcggcgagc ggtatcagct cactcaaagg cggtaatacg gttatccaca
      781 gaatcagggg ataacgcagg aaagaacatg tgagcaaaag gccagcaaaa ggccaggaac
      841 cgtaaaaagg ccgcgttgct ggcgtttttc cataggctcc gcccccctga cgagcatcac
      901 aaaaatcgac gctcaagtca gaggtggcga aacccgacag gactataaag ataccaggcg
      961 tttccccctg gaagctccct cgtgcgctct cctgttccga ccctgccgct taccggatac
     1021 ctgtccgcct ttctcccttc gggaagcgtg gcgctttctc atagctcacg ctgtaggtat
     1081 ctcagttcgg tgtaggtcgt tcgctccaag ctgggctgtg tgcacgaacc ccccgttcag
     1141 cccgaccgct gcgccttatc cggtaactat cgtcttgagt ccaacccggt aagacacgac
     1201 ttatcgccac tggcagcagc cactggtaac aggattagca gagcgaggta tgtaggcggt
     1261 gctacagagt tcttgaagtg ggtggcctaa ctacggctac actagaagaac agtatttggt
     1321 atctgcgctc tgctgaagcc agttaccttc ggaaaaagag ttggtagctc ttgatccggc
     1381 aaacaaacca ccgctggtag cggtggtttt tttgtttgca agcagcagat tacgcgcaga
     1441 aaaaaggat ctcaagaaga tcctttgatc ttttctacgg ggtctgacgc tcagtggaac
     1501 gaaaactcac gttaagggat tttggtcatg agattatcaa aaaggatctt cacctagatc
     1561 cttttaaatt aaaaatgaag ttttaaatca atctaaagta tatatgagta aacttggtct
     1621 gacagttacc aatgcttaat cagtgaggca cctatctcag cgatctgtct atttcgttca
     1681 tccatagttg cctgactccc cgtcgtgtag ataactacga tacgggaggg cttaccatct
     1741 ggcccagtg ctgcaatgat accgcgagac ccacgctcac cggctccaga tttatcagca
     1801 ataaaccagc cagccggaag ggccgagcgc agaagtggtc ctgcaacttt atccgcctcc
     1861 atccagtcta ttaattgttg ccgggaagct agagtaagta gttcgccagt taatagtttg
     1921 cgcaacgttg ttgccattgc tacaggcatc gtggtgtcac gctcgtcgtt tggtatggct
     1981 tcattcagct ccggttccca acgatcaagg cgagttacat gatcccccat gttgtgcaaa
     2041 aaagcggtta gctccttcgg tcctccgatc gttgtcagaa gtaagttggc cgcagtgtta
     2101 tcactcatgg ttatggcagc actgcataat tctcttactg tcatgccatc cgtaagatgc
     2161 ttttctgtga ctggtgagta ctcaaccaag tcattctgag aatagtgtat gcggcgaccg
     2221 agttgctctt gcccggcgtc aatacgggat aataccgcgc cacatagcag aacttttaaaa
     2281 gtgctcatca ttggaaaacg ttcttcgggg cgaaaactct caaggatctt caggtgatcg
     2341 agatccagtt cgatgtaacc cactcgtgca cccaactgat cttcagcatc ttttactttc
     2401 accagcgttt ctgggtgagc aaaaacagga aggcaaaatg ccgcaaaaaa gggaataagg
     2461 gcgacacgga aatgttgaat actcatactc ttccttttc aatattattg aagcatttat
     2521 cagggttatt gtctcatgag cggatacata tttgaatgta tttagaaaaa taaacaaata
     2581 ggggttccgc gcacatttcc ccgaaaagtg ccacctgacg tctaagaaac cattattatc
     2641 atgacattaa cctataaaaa taggcgtatc acgaggccct ttcgtc
//
```

Figure 2: Sample genbank (.gb) file

4.2 Signature identification

In a digitally signed document, the original message and the signature can be easily identified and separated since we have delimiters that separate them. In the DNA domain, there exists another problem of embedding the signature inside the original molecule. Because the site of insertion will vary depending on the architecture of the plasmid, delimiters are needed to identify where the

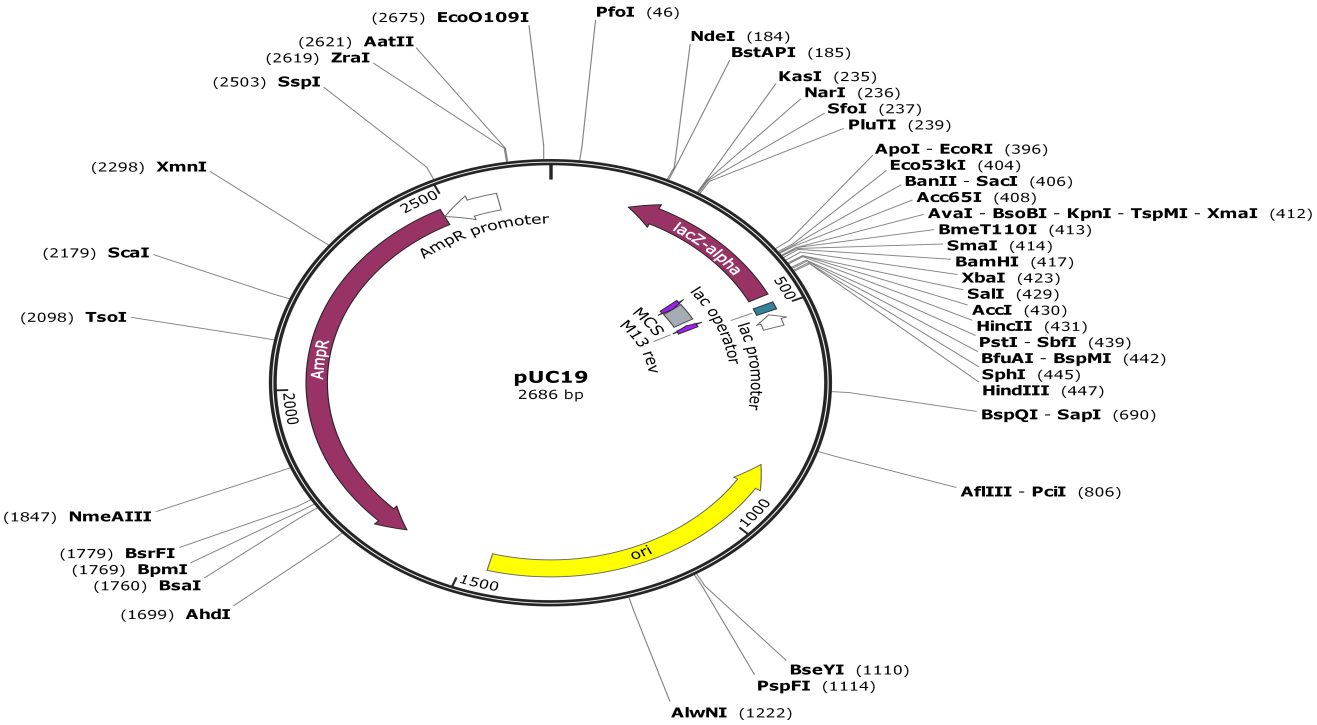

Figure 3: SnapGene view of a genbank file

signature starts and ends. So we have used an algorithm that identifies subsequences. Any subsequence of 10 base pairs (substring of length 10) that is not present in the original sequence can be used as a start and end delimiter which will contain the signature. During verification, all subsequences of 10 base pairs will be identified and only those subsequence that occur twice within the entire sequence are the delimiters.

Although the above technique can be useful, to keep things simple we came up with an alternate solution. Instead of the algorithm choosing the delimiters, the tool we developed lets the user input their own delimiters of 10 base pairs. This approach can be beneficial as it lets the biologist design delimiters that are relevant to their specific project. For instance, the delimiters can be designed in such a way as to simplify synthesis/assembly of the DNA. The tool checks if the sequences are permitted i.e. the 10 base pair subsequence does not already exist elsewhere in the plasmid. The sequences that we have used as start and end delimiters are ACGCTTCGCA and GTATCCTATG respectively. These sequences are relatively easy to identify visually, they are unlikely to develop secondary structures and they contain a balanced number of A's C's G's and T's.

4.3 Error tolerance

When any digitally signed message is shared and verification fails, the sender just resends the message again. But in the domain of DNA sharing, since we are primarily shipping samples, this implies resending and likely resynthesizing the sample (sometimes even batches of samples). This will incur a lot of cost. The presence of a signature inside the molecule will ensure that any change in the signed DNA will result in failed verification. But DNA molecules are prone to naturally occurring mutations. Hence after a failed verification, it can be useful to check the location of the mutation(s) which caused the verification to fail. If there are mutations in any important features, the receiver will likely choose to reorder the sample. If there are mutations in any relatively unimportant part of the DNA, the receiver may choose to proceed to work with the sample. In order to achieve this error tolerance, we have used error correction codes specifically Reed-Solomon Codes [14]. The details are described in section 7.

4.4 Strong association between physical DNA molecule and its digital representation

In order to tie the physical DNA sample with its digital representation, we have proposed a solution that combines the signed sequence and its description and generates a signature on this combined message. This signature is placed in the digital representation of the DNA such as the genbank file which is shared with the receiver. This ensures that the explanation of the sequences and the sequences in the plasmid are correct and related. Any change in the descriptions without changing the molecule will invalidate this signature. Also, any change in the molecule without updating the descriptions will invalidate the signature. The details of this procedure are described in section 9.

5 DNA SIGNATURE SCHEME

We use the identity-based signature scheme proposed by Shamir[16] in 1984. However, we have simplified it to suit the DNA sharing domain. The unique identifier in our case is the ORCID (Open Researcher and Contributor ID). Shamir's IBS is based on the RSA cryptosystem and its security depends on the difficulty of integer factorization in the RSA problem.

Setup: The setup is similar to the standard RSA cryptosystem setup. For a given security parameter k, proceed with the following steps -

(1) Generate two distinct primes p and q at random with $2^{\frac{k-1}{2}} < p, q < 2^{\frac{k}{2}}$
(2) Calculate the modulus n as $n = p \cdot q$
(3) Calculate the totient $\phi(n) = (p-1)(q-1)$. Choose the master public key e as $1 < e < \phi(n)$, such that e is relatively prime to $\phi(n)$.
(4) Calculate the master private key, d, as $e^{-1}\ mod\ \phi(n)$ in order to satisfy the congruent relation $d \cdot e \equiv 1\ mod\ \phi(n)$
(5) Publish the public parameters $<e, n>$ and the keep the private key d.

In our setup k is 1024 bits.

Key Extraction: The private key, s_{ID} for a user with the identity ID is generated as:

$$s_{ID} = H(ID)^d\ mod\ n$$

where H is a secure hash function. We are using SHA-256 as the hash function.

Signature Generation: For generating the signature for a message $m \in \{0,1\}^*$, generate the signature(σ) as :

$$\sigma = s_{ID}^{H(m)}\ mod\ n = H(ID)^{d \cdot H(m)}\ mod\ n$$

Signature Verification: To verify a signature σ for a message m and user identity ID, check if the following equation holds:

$$\sigma^e \stackrel{?}{=} H(ID)^{H(m)}\ mod\ n$$

Proof of security for DNA signature scheme

Our DNA signature scheme is a simplified version of Shamir's IBS scheme. To show that our scheme is secure we first present the original Shamir's IBS scheme:

Shamir's IBS Scheme

Setup: Same as above.

Key Extraction: Same as above.

Signature Generation: For generating the signature for a message $m \in \{0,1\}^*$:

(1) Choose $r \in_R \mathbb{Z}_n^*$.
(2) Compute $R = r^e\ mod\ n$.
(3) Compute $c = H(R||m)\ mod\ n$.
(4) Compute $t = s_{ID} \cdot r^c\ mod\ n$.
(5) Output signature $\sigma = (R, t)$

Signature Verification: To verify a signature σ for a message m and user identity ID, check if the following equation holds:

$$t^e \stackrel{?}{=} H(ID) \cdot R^{H(R||m)}\ mod\ n$$

We show that the simplified Shamir's IBS scheme is able to provide the same level of security guarantees as the original scheme.

The signature in the original scheme is a tuple - (R, t). If the modulus chosen is 1024 bits, the signature output will be 2048 bits which is 1024 base pairs. Based on our threat model, Shamir's IBS scheme is secure if no polynomial-time adversary can forge the signature on a given message. It is readily shown that this is equivalent to the difficulty of breaking the RSA public-key cryptography. Here is how. To forge a signature, the adversary needs to find s_{ID} from the equation $t = s_{ID} \cdot r^c\ mod\ n$. Let, $r^c = w$. Therefore, $s_{ID} = t \cdot w^{-1}$. In order to find any inverse modulo n, one has to know $\phi(n)$, where $\phi(\cdot)$ is the Euler totient function. Calculating $\phi(n)$ from n is equivalent to factoring n into two distinct primes – a known hard problem. Next, to calculate w^{-1}, the random r has to be calculated. If r can be found, then r^c can be found as c is public. $c = H(R||m)\ mod\ n$. R is first part of the signature and m is the message which bears the signature. To find the random r, one has to know $\phi(n)$ or the secret key d, since $R = r^e$, $r = R^d$.

With that brief background, let us now consider the simplified Shamir's IBS scheme. We have simplified the original scheme by removing the random R. In our scheme, the signature $\sigma = s_{ID}^{H(m)}$. Therefore, $s_{ID} = \sigma^y$, where $y = H(m)^{-1}$. Hence to find y, one has to know $\phi(n)$ which is equivalent to the RSA problem. Therefore, no polynomial-time adversary can forge a signature in the simplified scheme.

Note that our scheme will generate the same signature for the same message every time. This is a threat where replay attacks on signatures is of concern. In our domain, the threat of replay attacks is negligible since replaying the signed message implies sending the actual signed DNA to the receiver again. As this is not any digital message which can be generated by packet crafting or similar techniques, the attacker would have to actually synthesize the DNA molecule and send it to the receiver. On the other hand, removing the random will make the signature length 1024 bits or 512 base pairs if the modulus(n) is 1024 bits. Hence, in our domain, although the same original DNA plasmid, originating from the same source, will have the same signature every time, the practical risk is minimal and is outweighed by the benefit of minimizing the signature length so as to decrease the likelihood that the functionality or stability of the plasmid is disrupted. □

Figure 4 depicts a map of the plasmid features of a digitally signed DNA file. We are exploring other identity based signature schemes which generate shorter signatures as the signature sequences have to be ordered from gene synthesis companies and this can be financially expensive. The cost of synthesis is about 7 to 9 cents per base pair.

6 SIGN-SHARE-VALIDATE WORKFLOW

In our system, there are three players: 1) The signer will develop the DNA signature and sign a sequence. 2) The verifier will use the signature to verify whether the received DNA sequence was sent by the appropriate sender and was unchanged after signing. 3) A Central Authority will provide the signer with a token that is associated with their identity. We assume that the central authority is secure and trusted by all participants in the system.

There are also three steps to the sign-share-validate workflow, summarized in Figure 5. In this example, Alice is developing a new plasmid. She starts in a sequence editor application by combining

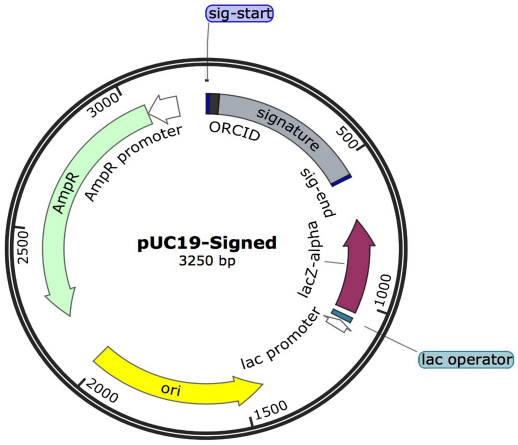

Figure 4: Snapgene view of a signed digital DNA file.

sequences from different sources. When she has finalized the sequence of the plasmid she wants to assemble in the lab, she uses the signature generating service hosted in a server to create a DNA signature sequence she will add to her design. This DNA sequence is the digital signature. It is generated using the signature algorithm described in section 5. Alice provides the digital DNA file to sign, her unique identifier (ORCID) and a six-digit plasmid ID. The digital signature is inserted in the plasmid sequence between two conserved sequences used to identify the signature from the rest of the plasmid sequence. Alice will then assemble the signed plasmid by combining DNA fragments from different sources. She will have to order the DNA fragment corresponding to the signature from a gene synthesis company. She describes her plasmid in a paper and refers to it using the six-digit number ID which she used to identify the plasmid in the signature. She did not include the entire plasmid sequence in the online supplement of the article (this is the norm for biology publications). She sends the plasmids to a few collaborators.

Ellen is interested in using Alice's plasmid. She gets the plasmid from another graduate student who got it from his advisor a few years ago. Ellen has limited confidence in the plasmid because it came in a hand-labeled tube. So, she decides to get it sequenced completely before doing anything with it. She uploads the assembled sequence of the plasmid to the server to verify the plasmid. The signature validation service in the server identifies the signature inserted between the two signature tags. It will identify a block of 32 bp to the right of the signature and extract the plasmid developer's ORCID. Using the ORCID value as identity, the server decrypts the 512 bp signature block. Then, the validation service will verify the signature as described in section 5. If the two values match, then Ellen will know that the plasmid was signed by Alice and that the physical sequence of her plasmid corresponds exactly to Alice's design. She had asked Alice for the plasmid sequence to align with her sequencing data. Unfortunately, Alice had moved on with her life and she no longer had access to the plasmid sequence files. Nonetheless, because she was careful enough to sign her plasmid, Ellen can be assured that the plasmid she intends to use is the one described in the publication.

Alternatively, the validation service might have determined that the signature was invalid. Several hypotheses could lead to this situation. It is possible that Alice was sloppy and did not manage to assemble the plasmid corresponding to the sequence she had designed. It is also possible that her advisor handed Ellen a derivative of the plasmid described by Alice. One could also not rule out the possibility of spontaneous mutations or a stupid labeling error. In this situation, Ellen may decide to proceed with the plasmid based on the similarity of the plasmid sequence and the information available in the Methods section of the paper describing the plasmid.

7 ERROR CORRECTION CODES

Limitations of using signed DNA: The presence of a digital signature within a plasmid will guarantee that the original sequence, identity sequence and the signature sequence are unchanged since the time the plasmid was signed. If any of these change, intentionally or unintentionally, the receiver will not be able to verify the DNA sequence. DNA is prone to mutations. Mutations are a naturally occurring phenomenon. Whenever there is any mutation within the signed DNA, the receiver will not know what changes occurred to invalidate the signature.

Mutations can be of three types: 1) **Point mutation** - This is the case where one base changes to another base e.g. AAGGAA $\longrightarrow$ AAGAAA. 2) **Insertion** - This is when a subsequence gets added to the original sequence e.g. AAGGAA $\longrightarrow$ AAGAGAA. 3) **Deletion** - This is when a subsequence gets deleted from the original sequence. e.g. AAGGAA $\longrightarrow$ AAGG. In any of the above scenarios, the verification process will result in failure. In the digital realm, if any message is not verified, we can always resend the message. But in the DNA sharing domain, this requires that the sample is transported and/or synthesized again, which incurs a lot of cost. Associated with the problem of mutation lies the problem of sequencing. When the DNA is processed by an automated DNA sequencer, the output is not always one hundred percent correct. It is dependent on the depth of sequencing, and increased sequencing depth means higher costs. Sequencing a small plasmid to sufficient depth is relatively inexpensive, but for larger sequences, sequencing errors can be an issue.

In order to overcome these limitations, we propose using error correction codes along with signatures. The presence of error correction codes will help the receiver to locate a limited number of errors in the sequence. In information theory and coding theory, error detection and correction are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow such errors to be detected, and error correction is used to reconstruct the original, error-free data. In error correction, redundant data, or parity data, is added to a message, such that it can be recovered by a receiver even with a number of errors (up to the capability of the code being used). Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, and hard disks. It is possible to use the same techniques to provide a reliable reconstruction of sequences provided a small

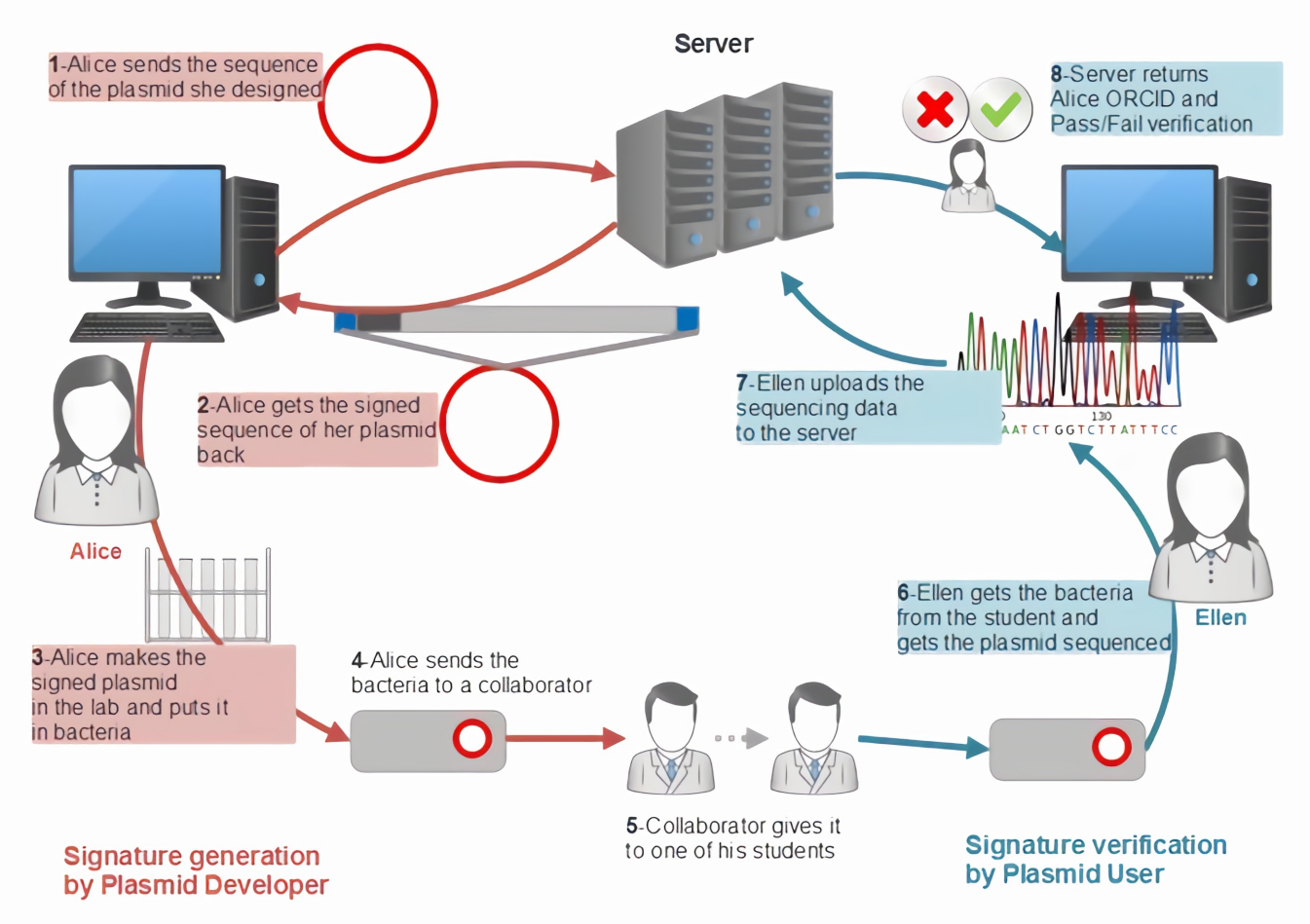

Figure 5: Example of a sign-share-validate workflow

number of changes have occurred. Our application of error correction codes to DNA can also be used to ensure the integrity of digital information stored in DNA molecules.

We have used Reed-Solomon codes [14] for error detection of DNA sequences. As of now, we have utilized this approach to correct point mutations only. Reed-Solomon codes are block-based error correcting codes with a wide range of applications in digital communications and storage. They are used extensively in storage devices e.g. CD/DVD, barcodes, QR codes, wireless communication, satellite communication (voyager space probe used RS codes) etc. We will not explain the construction and working principle of a Reed-Solomon code here. An interested reader can refer to [13] for details.

The most common convention of a Reed-Solomon code is (255, 223, 32) in which 223 is the number of data symbols, 32 is the parity symbols and 255 is the total number of symbols that can be processed at a time or block size. Using this convention the total number of errors that can be corrected anywhere in the 255 symbols is 32/2 = 16. This convention uses 8-bit symbols. Since the symbols are 8-bits, the block size is $2^8 - 1 = 255$ and with respect to a programming language, each symbol is treated as a byte. So the Reed-Solomon code of (255,223,32) can be simply put as 255 bytes block size, 223 data bytes, and 32 parity bytes. The total number of errors that can be corrected is 16 bytes. The parameters are generated from Galois field GF(257).

Here also we face an obstacle while applying error correction codes to DNA. Note that a plasmid contains between 2500 to 20,000 base pairs. If we treat each base as a character or byte, we will not be able to process the entire sequence in a single block. We have to make blocks of 255 bases. This implies that in a block of 255 bases, 223 bases are the actual sequence and 32 bases are the parity sequence for that block. As we mentioned earlier, we do not have the privilege of using any delimiters in DNA as we had in a digital message. Therefore, the parity sequences of every block cannot be identified. Also, let us consider the following scenario - a user wants to correct 5 bases in a plasmid which contains a total of 800 bases. By convention, 800 bases cannot be processed at a time and will be processed in four blocks (255 bases in each). We cannot be certain about the distribution of errors in the four blocks i.e. there is no guarantee where the 5 errors might be. They may

be all in one block or distributed across multiple blocks, but the distribution will most certainly not be uniform (1.25 in each block). So assuming the worst case, we need to correct 5 errors in each of the blocks. Therefore the number of parity bytes for total 800 bases is now $10 \cdot 4 = 40$. Whereas, if we could process the entire 800 base sequence at once, we would have only $5 \cdot 2 = 10$ parity bytes.

To adapt to this scenario, we used 16-bit symbols or shorts. Now the block size is $2^{16} - 1 = 65535$. The sequence characters, which were bytes, are now shorts. The parameters are generated using Galois field GF(65537). This gives us the flexibility to process the entire plasmid sequence at once. The user provides the number of errors that they would like to detect. The original plasmid sequence and the generated sequence are passed to the Reed-Solomon encoder. The Reed-Solomon encoder generates $2 \cdot k$ shorts for k error tolerance. The $2k$ shorts are then converted to sequences. Each parity short is converted to an 8 base sequence (16/2). Previously the final signature sequence consisted of - $< start >< ORCID + Plasmid\_ID + Signature >< end >$ where start and end were 10 base pairs each, ORCID was 32 base pairs, Plasmid_ID was 12 base pairs and Signature was 512 base pairs. Now the parity sequences are inserted between the signature sequence and end sequence. Updated signature sequence - $< start >< ORCID + Plasmid\_ID + Signature + Parity >< end >$. During the correction phase, the parity sequence is retrieved using the start and end sequences and the length of the other three parts which is already known. The number of errors that can be corrected can be determined by the length of the parity sequence. Since each parity short is 8 bases, 16 bases are two shorts and two shorts can correct one error, hence the number of errors that can be corrected are - (parity sequence length) / 16.

Using the error correction code, the verifier can correct some number of errors (limit is set by signer) in the digital sequence file. Upon correcting the digital sequence, the verification is invoked again on the corrected sequence. The position of the errors and the corrected value are conveyed to the verifier. The verifier can then decide if the errors are in any valuable feature or not. If a valuable feature has been corrupted, the verifier can ask for a new shipment, else if the error was in a non-valuable area in the plasmid, the verifier can proceed to work with it.

The sign-share-verify workflow will be updated as follows. Ellen uploads the digital DNA file to the server which she obtained after sequencing the plasmid shared by Alice. The validation service tries to validate the sequence. If this validation results in failure, the error correction part is invoked and tries to correct the sequence depending on the number of errors Alice chose to tolerate during signing. If no corrections can be made (because the number of mutations in the sample exceeds the threshold set by Alice) Ellen is notified with an alert. If corrections can be made, the verification starts again on the corrected sequence. Upon successful verification on the corrected sequence, Ellen is notified about the errors (mutations) that occurred in the sample she received.

8 SIGNATURE GENERATION AND VERIFICATION PROCEDURE

The prototype tool we have developed allows a user to generate and validate signatures. The parameters i.e. e, d, N are fixed in the prototype where the modulus N is 1024 bits.

Signature generation: The user provides the following inputs for signature generation -

(1) The genbank (.gb) file.
(2) ORCID - a 16 digit number in xxxx-xxxx-xxxx-xxxx format.
(3) Plasmid ID - a 6 digit number.
(4) Location of signature placement.
(5) Number of errors to be tolerated.

All the necessary input checks e.g. the file has extension .gb, ORCID format is correct, ORCID is integers etc. are done. The signature generation procedure begins by splitting the genbank file by the keyword *ORIGIN*. Refer to figure **??**. After the keyword *ORIGIN* is the actual sequence and before it are the descriptions. The sequence is the message to sign and the descriptions are kept for verifying if the user provided location is colliding with an existing feature. Let us assume the sequence to sign is **SEQUENCE** and there exists a feature from location 1 to 3 which corresponds to **SEQ**. Next,the location of signature placement is checked. If the location collides with a feature, the user is alerted to change the location. For our example, if the user had provided 2, the tool will alert the user that there is already a feature **SEQ** there and ask for a new location. If the user chooses 4 which is after the letter **Q**, it will be allowed. Next, the OCRID and Plasmid ID are converted to ACGT sequence by the following conversion method - [0 - AC, 1 - AG, 2 - AT, 3 - CA, 4 - CG, 5 - CT, 6 - GA, 7 - GC, 8 - GT, 9 - TA]. The reason for choosing this conversion type is that if any ORCID or plasmid ID has repetitions e.g. 0000-0001-4578-9987, the converted sequence will not have a long run of a single base. If we used 0 - AA, the example ORCID would have AAAAAAAAAAAAAA in the beginning, and long runs of a single nucleotide can result in errors during sequencing. In the chosen conversion method the ORCID would start with ACACACACACACAC. Let the converted ORCID and Plasmid ID sequence be **ORCID** and **PID**. The signature is generated according to the scheme described in section 5. The signature bits are then converted to ACGT sequence. Let this signature sequence be **SIGN**. Also, recall that the start and end tag where this signature is to be placed is predefined. Let this start tag be **START** and end tag be **END**. The signature sequence is concatenated with ORCID and plasmid ID and then placed between the start and end - **START ORCID PID SIGN END**. This entire string is placed at the position specified by the user. As we chose 4 in our example, the total sequence looks like - **SEQ START ORCID PID SIGN END UENCE**. Now this string is passed into the error correction encoder. According to the number of tolerable errors specified by the user, the parity bits are generated. The parity bits increase with the number of errors to be tolerated. These parity bits are then converted to ACGT sequence. Let this be **ECC**. When the encoder output is generated, the string looks like - **SEQ START ORCID PID SIGN END UENCE ECC**. Next, the **ECC** is separated and this is placed before the signature and end tag. So the final output string is - **SEQ START ORCID PID SIGN ECC END UENCE** . Note that the error correction code is generated after generating the

signature sequence and combining with original sequence. Hence any error in that string can be corrected provided it is within the tolerable limit. For our example, if we put 2 as our error tolerance limit, then any 2 errors within the string **SEQ START ORCID PID SIGN END UENCE ECC** can be tolerated. For example if there is 1 error in **SEQ** and 1 error in **SIGN**, or 2 errors in **SIGN**, or 1 error in **SIGN** and 1 error in **ECC**, it can be corrected. But if there are more than two errors it cannot be tolerated. The final output string - **SEQ START ORCID PID SIGN ECC END UENCE** is written into another genbank file. The descriptions are updated i.e. the locations of the signature, start, end, ecc are added and if there are features after the signature placement locations they are updated. The output genbank file is shared with the recipient.

Signature verification: The user provides the following inputs for signature verification -

(1) The shared genbank (.gb) file.
(2) The fasta (`.fasta`) file which the receiver obtained after sequencing the shared, signed DNA.

The sequence in the fasta file might not be the in the same order as the receiver sent it. That is, after sequencing the shared DNA, the fasta file may look like - **ORCID PID SIGN ECC END UENCE SEQ START** which is a cyclic permutation of the final sequence the receiver obtained after signature generation. The genbank file contains the correct order. The tool aligns the genbank sequence and the fasta sequence. If there is any mutation in the shared DNA the fasta file will have some errors but most of it will be aligned correctly. If there are no mutations the file will be aligned perfectly. For now let us assume there is no mutation and hence the fasta sequence and the genbank sequence will be aligned perfectly - **SEQ START ORCID PID SIGN ECC END UENCE**. The tool looks for start and end tags which we had predefined. After obtaining the start tag, 32 bases are counted, this is the ORCID sequence, next 12 bases are counted, this is the plasmid ID sequence, then 512 bases are counted, this is the signature sequence. Next the substring after this signature sequence to the end tag is retrieved, this is the error correction sequence. Finally, the portion before start tag and the portion after end tag is concatenated to reconstruct the message for signature verification. So as of now we have retrieved **SEQUENCE**, **START**, **ORCID**, **PID**, **SIGN**, **ECC** and **END**. The **SEQUENCE**, **ORCID** and **SIGN** is used for signature verification according to the scheme described in section 5. If there is no mutation, the signature verification will succeed and the user is alerted for successful verification. If there is any mutation ,the verification will fail. In this case the extracted parts are used to construct the string - **SEQ START ORCID PID SIGN END UENCE ECC**. Recall that this was the output of the error correction encoder. If the error is within the tolerable limit, it will be corrected. If the error is more than the tolerable limit, the user is alerted that the verification and the error correction both failed. If the error is corrected, we again use the counting method to retrieve the corrected parts - **SEQUENCE**, **START**, **ORCID**, **PID**, **SIGN**, **ECC** and **END**. The verification is invoked on the corrected **SEQUENCE**, **ORCID** and **SIGN**. If the verification succeeds the second time the user is notified about success. Also, the corrected parts and the previously extracted parts (before first verification) are compared to display where the error was. If the verification fails on the corrected parts, the user is notified about failure after correction and the corrected errors are displayed.

It is to be noted that although the corrected errors are displayed to the user, the actual content of the DNA did not change, only the fasta representation changed. The physical DNA still contains the error i.e. if the sample is sequenced again, the freshly obtained fasta file will again be erroneous. Hence, the error correction works more like error detection. The user gets the correction information, and if he thinks the errors are not in any important part of the DNA, the user can choose to work with the shared sample. If he thinks the errors are in an important part of the DNA, the sample can be re-ordered from the sender.

9 ASSOCIATING THE DNA SAMPLE WITH ITS DIGITAL REPRESENTATION

When any researcher shares a DNA sample with another, he manually describes the additional features present in the sample in its digital representation (genbank file). This file can then be shared with the recipient (often, however, only the physical sample is shared, not the digital representation). The recipient will receive the sample, sequence it and obtain its digital representation. The common features that are in the sample can be automatically annotated with the help of a sequence-manipulation software like Snapgene. But the additional features and descriptions that Snapgene will not be able to interpret must be provided by the sender. In order to associate this digital DNA file which contains the additional descriptions(let us call this F_{sent}) with the digital DNA file that the receiver generates after sequencing the sample(let us call this F_{gen}), we associate them together with a combined signature. The association between these two files is created in the following way:

Create association

(1) Signer provides the digital DNA file containing the appropriate sequence and descriptions. Extract the sequence and the descriptions. Only the sequence is used for signature creation as described in section 5. Let this sequence be m_{seq} and the descriptions be m_{desc}.
(2) Generate signature on m_{seq} as before and place this within the original sequence. Let this final signed sequence be m_{sig}.
(3) Combine m_{desc} and m_{sig} by calculating the following:

$$m_{comb} = H(H(m_{sig})||H(m_{desc}))$$

where H is a secure hash function e.g. SHA-256 and $||$ is concatenation operation.
(4) Create signature for this m_{comb} using the same procedure -

$$\sigma' = s_{ID}^{m_{comb}} \ mod\ n = H(ID)^{d \cdot m_{comb}} \ mod\ n$$

(5) Add σ' to the genbank file with a keyword "ASSOC".
(6) Share the file with the recipient.

Validate association

(1) Recipient obtains F_{sent} and generates F_{gen} from the received sample. The tool takes both files as input.
(2) Extract σ', m_{desc} from F_{sent} and m_{sig} from $Fgen$. The ID is extracted from m_{sig}.

(3) Calculate m_{comb} as:

$$m_{comb} = H(H(m_{sig})||H(m_{desc}))$$

(4) Check if the following equation holds:

$$(\sigma')^e \stackrel{?}{=} H(ID)^{m_{comb}} \ mod \ n$$

Using this combined signature, the recipient can validate that the description file was sent by the authentic sender, the manually added descriptions have not been changed and these descriptions belong to the same DNA sample that was shared. The sign-share-verify workflow will work as follows. Ellen will upload the digital DNA file she generated after sequencing the sample shared by Alice and also the digital DNA file that Alice shared (which contains the additional descriptions). The server will match the combined signature and Ellen will be notified about the association between the two files.

10 DISCUSSION

The ability to digitally sign a synthesized DNA molecule allows rigorous tracking of a DNA molecule to its origin. It opens up the possibility of a number of novel applications.

One question that arises is, since additional base pairs must be added to the original DNA molecule to accommodate the signature and error correction feature, at what threshold can we add base pairs without affecting functionality? Unfortunately, there has been no concrete study as yet to determine the threshold of how many or what percentage of sequence addition retains the DNA's original properties and after what limit the properties break. Since a plasmid size ranges from about 2000 to 20,000 bases, it can be possible that a 512 base signature affects the properties of a 2000 base plasmid whereas it behaves perfectly fine on a 20,000 base plasmid. What we know so far is that the fewer the sequences are added to the molecule, the better chance there is to retain the properties. We have begun experiments to determine whether the 512 base pair signature plus error correction code (totaling 624 base pair) impacts the function of a 2868 base pair plasmid sequence in E. Coli.

Note that there is also an economic cost to adding base pairs to the synthesized DNA. After generating the signature sequence, it has to be synthesized by a gene synthesis company which involves cost, around 9 cents per base pair. Hence, it is essential to have a shorter signature length. We are trying to further shorten the signature size such that we can have better chances of property retention and provide cost-effectiveness.

Another aspect to consider is how to determine if a DNA molecule is signed or not since the delimiters are sequence specific. The solution is simple if the plasmid developer shares the start and end delimiters that have been used in signing the plasmid (as part of the annotated GenBank file or separately). The receiver can also look the plasmid up by its ID and the ORCID of the developer (possibly) in a public database maintained by a server. If the receiver knows the start and end delimiter sequences, the presence of the start and end delimiters can be verified using a number of biological methods:

(1) If the start and end delimiters match the recognition site of restriction enzymes (https://en.wikipedia.org/wiki/Restriction_enzyme), a restriction digest (https://en.wikipedia.org/wiki/Restriction_digest can be used to determine whether or not the sites are present.
(2) Chemical primers that are complementary to the start and end delimiters can be ordered and used in a diagnostic Polymerase Chain Reaction (PCR) experiment (https://en.wikipedia.org/wiki/Polymerase_chain_reaction).
(3) Radio-labeled probes complementary to the start and end sequences can be used to determine whether or not the sequences are present in a Southern Blot Assay (https://en.wikipedia.org/wiki/Southern_blot).

11 CONCLUSIONS AND FUTURE WORK

In this paper, we discussed a novel effort to embed digital signatures into DNA molecules that are synthesized in the laboratory. Do digital security solutions, in particular cryptographic techniques, map well to a physical world? That was the big question that we asked ourselves.

Synthetic DNA molecules are frequently shared physically. There is a need to bestow origin attribution properties to these molecules which are often licensed intellectual property. However, recent efforts to provide this property using watermarking techniques suffer from the problem that the watermark is independent of the DNA molecule (although it is embedded in the molecule). Thus, the watermark can potentially be removed from a physical DNA and embedded in another sample or replaced with another entity's watermark. We describe our efforts to provide more secure origin attribution properties using digital signatures.

As a proof of concept, signatures will be generated in the laboratory and inserted into two plasmids. The first, 401734, is a synthetic plasmid composed of two antibiotic resistance genes and an origin of replication. The second, 190691, is the commonly used standard vector pUC19. The physical signature embedding and validation protocol is a complex biological process, detailed discussion of which involves extensive domain knowledge beyond the scope of this work. Consequently, we omit this discussion from this paper. We are also still in the process of performing additional validation related experiments, results of which will be published as they become available.

There are several research questions that have not yet been answered and works that we plan to undertake in this area. First, our choices of parameters for the digital signatures as well as error correction codes are not based on some mathematical models of properties of the DNA vis-a-vis its size. Rather they are based on domain knowledge and experimental analysis. Models of DNA properties can help in making optimal parameter choices to trade off the size of the sequences encoding signatures, security strengths of signatures and degree of error resiliency.

Related to this first research challenge is the question of whether embedding signatures is applicable for naturally occurring DNA and for DNA sequences larger than plasmids such as microbial or even plant or animal genomes. We have assumed that there is some leeway regarding the properties of DNA molecules that allows for the addition of a signature without altering function or stability. Current knowledge indicates that large portions of genomes probably do not play a functional role. So it is possible to embed a

signature in these regions. However, from an evolutionary standpoint, sequences which do not play a functional role are less likely to be retained unchanged in the genome. Additionally, the sheer size of a genome means that mutations are more likely to occur and sequencing technologies are not yet robust enough to provide completely error-free whole genome sequences. In that case, what kind of security guarantees can we provide? For instance, could we apply digital signatures only to portions of larger sequences that are significant, such as those parts of a microbial genome which have been re-engineered for a specific purpose? Are there alternate signature schemes that are worth investigating that would be better-suited to whole genomes?

A third research area would involve signing and verifying the same DNA molecule multiple times by different users. Alice signs and sends a DNA sample to Bob and Bob validates Alice's DNA. Then Bob continues to modify it, signs it and sends it to Mallory. Can Mallory only verify Bob's signature, or is there a way for Mallory to track the entire pathway starting from Alice? It would be interesting to see if the concept of aggregate signatures can be applicable in these scenarios. Also, it would be interesting to see if we put a signature on top of an existing signature whether the characteristic of the DNA changes or not. If it does not, how many signatures can be inserted before the characteristics of the original DNA molecule begin to change? Also, if we cannot put multiple signatures within the same DNA molecule, how do we remove the signature that was present before signing it again. Finally, does removing the signature also alters the property of the DNA?

A fourth problem that comes into focus is the issue of key revocation. What are the semantics of key revocation and how is this going to be achieved? Certainly key revocation in the cyber-bio world is very different from the digital world. If a key is revoked in the digital world, one can dis-validate any message signed with the revoked key. Signing, a duplicate copy of the same message is not expensive. However, it is not trivial to dis-validate a DNA molecule, synthesize a new one with the same properties and sign it with a new key. This causes us to rethink the whole key revocation model in the cyber-bio world.

A fifth research area involves studying the synthetic DNA sharing supply chain and developing techniques to ensure origin attribution. To illustrate the problem, recall that DNA molecules are shared to study their properties and possibly manipulate them to build a different DNA molecule. If a DNA molecule is licensed as an IP and is subsequently used in this manner, the synthesizer of the original molecule would like to get some credit for the root molecule. It will be interesting to study how if at all, this can be achieved.

ACKNOWLEDGMENT

This work is partly based on research supported by the Office of the Vice President of Research, Colorado State University. This material is also based upon work performed by Indrajit Ray while serving at the National Science Foundation. Research findings presented here and opinions expressed are solely those of the authors and in no way reflect the opinions of Colorado State University, the U.S. NSF or any other federal agencies.

REFERENCES

[1] George M. Church, Yuan Gao, and Sriram Kosuri. 2012. Next-Generation Digital Information Storage in DNA. *Science* 337, 6102 (2012), 1628–1628. https://doi.org/10.1126/science.1226355 arXiv:http://science.sciencemag.org/content/337/6102/1628.full.pdf

[2] Yaniv Erlich and Dina Zielinski. 2017. DNA Fountain Enables a Robust and Efficient Storage Architecture. *Science* 355, 6328 (2017), 950–954. https://doi.org/10.1126/science.aaj2038 arXiv:http://science.sciencemag.org/content/355/6328/950.full.pdf

[3] Daniel G. Gibson, John I. Glass, Carole Lartigue, Vladimir N. Noskov, Ray-Yuan Chuang, Mikkel A. Algire, Gwynedd A. Benders, Michael G. Montague, Li Ma, Monzia M. Moodie, Chuck Merryman, Sanjay Vashee, Radha Krishnakumar, Nacyra Assad-Garcia, Cynthia Andrews-Pfannkoch, Evgeniya A. Denisova, Lei Young, Zhi-Qing Qi, Thomas H. Segall-Shapiro, Christopher H. Calvey, Prashanth P. Parmar, Clyde A. Hutchison, Hamilton O. Smith, and J. Craig Venter. 2010. Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome. *Science* 329, 5987 (2010), 52–56. https://doi.org/10.1126/science.1190719 arXiv:http://science.sciencemag.org/content/329/5987/52.full.pdf

[4] Dominik Heider and Angelika Barnekow. 2007. DNA-based Watermarks Using the DNA-Crypt Algorithm. *BMC Bioinformatics* 8, 1 (29 May 2007), 176. https://doi.org/10.1186/1471-2105-8-176

[5] Clyde A. Hutchison, Ray-Yuan Chuang, Vladimir N. Noskov, Nacyra Assad-Garcia, Thomas J. Deerinck, Mark H. Ellisman, John Gill, Krishna Kannan, Bogumil J. Karas, Li Ma, James F. Pelletier, Zhi-Qing Qi, R. Alexander Richter, Elizabeth A. Strychalski, Lijie Sun, Yo Suzuki, Billyana Tsvetanova, Kim S. Wise, Hamilton O. Smith, John I. Glass, Chuck Merryman, Daniel G. Gibson, and J. Craig Venter. 2016. Design and Synthesis of a Minimal Bacterial Genome. *Science* 351, 6280 (2016). https://doi.org/10.1126/science.aad6253 arXiv:http://science.sciencemag.org/content/351/6280/aad6253.full.pdf

[6] Daniel C. Jupiter, Thomas A. Ficht, James Samuel, Qing-Ming Qin, and Paul de Figueiredo. 2010. DNA Watermarking of Infectious Agents: Progress and Prospects. *PLOS Pathogens* 6, 6 (06 2010), 1–3. https://doi.org/10.1371/journal.ppat.1000950

[7] Michael Liss, Daniela Daubert, Kathrin Brunner, Kristina Kliche, Ulrich Hammes, Andreas Leiherer, and Ralf Wagner. 2012. Embedding Permanent Watermarks in Synthetic Genes. *PLOS ONE* 7, 8 (08 2012), 1–10. https://doi.org/10.1371/journal.pone.0042465

[8] Peter Ney, Karl Koscher, Lee Organick, Luis Ceze, and Tadayoshi Kohno. 2017. Computer Security, Privacy, and DNA Sequencing: Compromising Computers with Synthesized DNA, Privacy Leaks, and More. In *Proceedings of the 26th USENIX Security Symposium*. Vancouver, Canada.

[9] Jean Peccoud, J. Christopher Anderson, Deepak Chandran, Douglas Densmore, Michal Galdzicki, Matthew W. Lux, Cesar A. Rodriguez, Guy-Bart Stan, and Herbert M. Sauro. 2011. Essential Information for Synthetic DNA Sequences. *Nature Biotechnology* 29 (January 2011). http://dx.doi.org/10.1038/nbt.1753

[10] Jean Peccoud, J Christopher Anderson, Deepak Chandran, Douglas Densmore, Michal Galdzicki, Matthew W Lux, Cesar A Rodriguez, Guy-Bart Stan, and Herbert M Sauro. 2011. Essential information for synthetic DNA sequences. *Nature Biotechnology* 29 (Jan. 2011), 22. http://dx.doi.org/10.1038/nbt.1753

[11] Jean Peccoud, Megan F. Blauvelt, Yizhi Cai, Kristal L. Cooper, Oswald Crasta, Emily C. DeLalla, Clive Evans, Otto Folkerts, Blair M. Lyons, Shrinivasrao P. Mane, Rebecca Shelton, Matthew A. Sweede, and Sally A. Waldon. 2008. Targeted Development of Registries of Biological Parts. *PLOS ONE* 3, 7 (2008), 1–7. https://doi.org/10.1371/journal.pone.0002671

[12] Jean Peccoud, Jenna E. Gallegos, Randall Murch, Wallace G. Buchholz, and Sanjay Raman. 2018. Cyberbiosecurity: From Näive Trust to Risk Awareness. *Trends in Biotechnology* 36, 1 (2018), 4–7. https://doi.org/10.1016/j.tibtech.2017.10.012

[13] James S Plank et al. 1997. A Putorial on Reed-Solomon Coding for Fault-tolerance in RAID-like Systems. *Software Prac. Experience* 27, 9 (1997), 995–1012.

[14] Irving S. Reed and Gustave Solomon. 1960. Polynomial Codes over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics (SIAM)* 8, 2 (1960), 300–304.

[15] Sarah M. Richardson, Leslie A. Mitchell, Giovanni Stracquadanio, Kun Yang, Jessica S. Dymond, James E. DiCarlo, Dongwon Lee, Cheng Lai Victor Huang, Srinivasan Chandrasegaran, Yizhi Cai, Jef D. Boeke, and Joel S. Bader. 2017. Design of a Synthetic Yeast Genome. *Science* 355, 6329 (2017), 1040–1044. https://doi.org/10.1126/science.aaf4557 arXiv:http://science.sciencemag.org/content/355/6329/1040.full.pdf

[16] Adi Shamir. 1985. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology*, George Robert Blakley and David Chaum (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 47–53.

[17] O. Tornea and M. E. Borda. 2009. DNA Cryptographic Algorithms. In *Proceedings of the International Conference on Advancements of Medicine and Health Care through Technology*, Simona Vlad, Radu V. Ciupa, and Anca I. Nicu (Eds.). Romania, 223–226.