Mueller *et al. BMC Bioinformatics* (2016) 17:389
DOI 10.1186/s12859-016-1256-6

BMC Bioinformatics

RESEARCH ARTICLE Open Access

A covert authentication and security solution for GMOs

Siguna Mueller[1], Farhad Jafari[2] and Don Roth[1*]

Abstract

Background: Proliferation and expansion of security risks necessitates new measures to ensure authenticity and validation of GMOs. Watermarking and other cryptographic methods are available which conceal and recover the original signature, but in the process reveal the authentication information. In many scenarios watermarking and standard cryptographic methods are necessary but not sufficient and new, more advanced, cryptographic protocols are necessary.

Results: Herein, we present a new crypto protocol, that is applicable in broader settings, and embeds the authentication string indistinguishably from a random element in the signature space and the string is verified or denied without disclosing the actual signature. Results show that in a nucleotide string of 1000, the algorithm gives a correlation of 0.98 or higher between the distribution of the codon and that of *E. coli*, making the signature virtually invisible.

Conclusions: This algorithm may be used to securely authenticate and validate GMOs without disclosing the actual signature. While this protocol uses watermarking, its novelty is in use of more complex cryptographic techniques based on zero knowledge proofs to encode information.

Keywords: GMO security, Limits of watermarking, Zero knowledge proofs, Verifiable encryption for GMO

Background

The dramatically increasing worldwide utilization of genetically modified plants, animals and microbes (GMOs) presents challenges to ensure security, authenticity and validation of material goods and legal agreements. Similarly to the evolution witnessed in internet protocols, strategic focus is required to anticipate, track and address potential infringements of GMO security. It is imperative that unimpeachable protocols assure product ownership, provide data to track the product supply chain, and to preempt malicious attacks especially related to bioagents, such as weaponized anthrax spores. As GMOs are not tamper proof or invulnerable to outside attack, it is necessary to encode and embed cyber-security data within the GMO genome. An ideal GMO based security mechanism should provide a secure authentication process accessible to relevant parties without revealing the specific signature components to outside parties. Watermarking has been used extensively to establish authentication signatures that validate ownership by providing a mechanism to conceal and recover the required data necessary to authenticate the identification signature of the originator [1–4]. However, in watermarking applications, the identity of the authentication information is disclosed as validity is verified [5–11]. Although these methods would be useful under many scenarios, they are unacceptable in the context of sophisticated GMO security because they would fail under concerted attack based on malicious transfer and signature duplication. For example, Clelland et al. [5], although establishing significant message secrecy, does not protect the key decoding signatures after access by a third party. It also appears that their approach would be best suited to small message concealment because the DNA based message length must be similar to that of the background sonicated genomic DNA (ca 50-150 nucleotides). Ideally, the signature should not be transferrable, and should remain concealed even from third parties utilizing the cryptographic code. Integrating

*Correspondence: RothDon@uwyo.edu
[1]Department of Molecular Biology, University of Wyoming, 1000 E. University Ave, 82071 Laramie, WY, USA
Full list of author information is available at the end of the article

this sophisticated cryptographic modification is critical for sustained next generation security of GMOs. A focused adversary should only be able to speculate whether a given sequence of nucleotides is a random sequence or a valid signature sequence. Considering a scenario where an attacker might seek to modify a GMO or weaponized bioagent in order to adversely affect function, coercion of individuals having information components or even alignment of whole genome sequences from target and wild type stains via BLAST or other programs should have an extremely low probability of identifying and/or validating an authentic signature. Clearly any security approach also must align with prokaryotic and eukaryotic GMO development and application technologies.

In this paper we demonstrate a novel algorithm that addresses these concerns using advanced cryptographic techniques. For every signature string generated, the developer generates a populating subset with fake signatures (a random sequence of nucleotides of the same length as a valid signature). Thus, each clone would contain adequate information for identification whether or not it is a signature carrying clone. The valid signature can be established by techniques developed in [12] and would be available to the developer and/or third party inspector. Those techniques assure that the input information is correctly received on the output side, i.e. when they are stored and read out there is no loss or change of information, while there is no concern that someone would actively modify or manipulate the data. Since each clone contains a valid or fake signature element, it would be virtually impossible to correctly select the authentic signature. We emphasize that while our algorithm uses watermarking, it is not a watermarking protocol per se. The process of watermarking by itself does not provide adequate security during its verification as it allows potentially malicious transfer and signature duplication. Our protocol uses sophisticated novel cryptanalytic attack models and protection mechanisms. While many of the existing methods use classic cryptography such as symmetric - AES, nonsymmetric -RSA and more classical approaches, we use zero-knowledge and confirmer signature techniques applied to GMOs. These are much more advanced crypto systems than AES and RSA. This algorithm should become the standard in implementation of this technology in practice.

Comparison with other work

Practical realization and combination with data-encoding mechanisms into DNA

Embedding of data in DNA has received a lot of attention. Previous algorithm proposals primarily concerned about biological aspects and correct and efficient decoding. Heider and Barnekow [13, 14] focus on error detection and correcting properties - not in the sense of cryptography - but inside the genome, to detect and correct mutations occurring during cell division that might destroy the information that is 'encrypted' (i.e., hidden), inside the genome. As such the DNA medium can be interpreted as a noisy channel and has been addressed by tools of digital coding and information theory [11]. Depending on usage requirements in living cells, our watermarking step might benefit from such additional features and could easily be combined with theirs or related algorithms. Yachie et al. [15] considered error detection and correction of the data-encoded DNA sequence inside of living organisms. Their approach is a refined repetition code that avoids multiple segments of the same DNA sequence within a single genome. Our method can be combined with their alignment-based DNA-data storage and retrieval method, or any of the sequence alignment methods. In fact, we propose a modern alignment method with provably secure decoding properties in [12].

However, no attention has been placed on the cryptographic aspect of the problem. In particular, in the case of ownership watermarks, biocompatibility along with the correct and efficient encoding is not enough. The embedded information that is stored and retrieved additionally requires specific cryptographic security requirements unique to this situation. It is imperative that the secret key remains hidden during watermark verification to prevent the unintended copying of signature data or specified information. These are independent cryptographic security considerations established in this paper.

Clarification about utilization of cryptography

In contrast to correctness of encoding and decoding and efficiency, cryptography considers the security aspects of a (digital) communication medium. These security concerns are unique to the DNA setting. An example can be seen work by Gibson et al. [16] who established the synthesis and assembly of a synthetically designed bacterial cell. Watermark sequences are included which distinguish the synthetically designed from naturally occurring DNA and cells. This type of watermark does not yield unique ownership in that watermark verification is only done by multiplex PCR and the entire watermarking procedure could be imitated by others. They were not concerned about cryptographic security features, e.g., if someone were to produce a harmful bacterial clone carrying their watermark information. How would they refute this clone was not theirs?

Haughton and Balado [11] first incorporated a secret k to keep the encoding secret. The key is shared only between an encoder and decoder. This has the advantage that only the encoder and decoder will have knowledge of the secret message that is embedded in DNA. In the context of ownership watermarking, unfortunately, this

scenario is not fully satisfactory. It requires that verification of the watermark is only possible to a selected list of decoders which has to be determined prior to embedding the watermark. Once the watermark is placed, the watermark verification process is only possible within this fixed set of users. In the case of verification of a watermark ownership to a user outside of this fixed set, this scenario is not applicable.

Heider and Barnekow [13] suggest to integrate several private and public key cryptographic algorithms, by employing encryption or a one-time pad. Both are done to create a short binary message. Although the authors did not make this explicit in their work, the first obvious advantage of this approach is that by doing so the information to be hidden inside the DNA is now scrambled inside a binary string. However, it is imperative to note that they do not utilize any of the mentioned cryptographic algorithms. What is needed for DNA-Crypt is a plaintext message (the information to be hidden inside the DNA) that is efficiently converted to a binary string. Thus, they correctly argue, that any function, mapping, or algorithm, which takes meaningful input and coverts it to binary, can be used for their purpose. Their main concern is only the output binary string. They do not incorporate any cryptographic features. They do not consider security, cryptographic approaches, or utilize encryption and decryption. In fact, they argue that the keys used for these cryptographic algorithms could be exchanged with other users. However, precisely for private key crypto, keeping the keys secret is the most important requirement to ensure security. Clearly, their concern is not to utilize the mentioned algorithms for their cryptographic features, but mainly to generate a binary string. Their concern is for better storage utilization, and hence, the cryptographic integration is only for compression purposes of text data into binary (source compression into binary). In summary, all that is utilized by DNA-Crypt is a binary encoding table [14].

In summary, while data embedding methods have benefited from numerous disciplines of digital communication theory, unique requirements of cryptography and security requirements are first addressed in this work. It is crucial to note, that our work can seamlessly be combined with previous data embedding methods. Balado-Haughton [17] determine the maximal number of ways that DNA watermarking can be done, by considering it as a special data hiding problem. Their basic requirement is the primary structure preservation achieved via the redundancy of the genetic code. This does not lead to a unique solution. Depending on biocompatibility constraints and other practical considerations, the tagging of DNA can be performed in various ways. We have not focused on length requirements of the signature sequence, how easily the signatures can be inserted and read, as our method can easily be combined with any others that focus on such issues.

Our work complements these proven practical realizations, can easily be combined with related successful in vivo experiments to hide the secret information in non-coding regions, and addresses the missing security issues not considered before.

Each of [11, 13–16] use some watermarking as a means to ensure tracking or ownership of DNA or organism. Table 1 gives a comparison with our work.

Methods

The basic cryptographic building block is defined as a zero-knowledge (ZK) proof of knowledge [18] of a hidden signature to constitute the designated confirmer signature [19]. ZK proofs are both convincing and yet yield no identifying key code information beyond the validity of the assertion being proved. They are typically used to force malicious parties to behave according to a predetermined protocol.

The specific cyber-security protocol is as follows: Let Σ be a digital signature scheme given by its key generation protocol Σ.keygen which generates a key pair Σ.sk and Σ.pk consisting of the secret and private key for the signature generation and verification protocols, respectively. Let Γ be a cryptosystem described by Γ.keygen that generates the pair (public key $=$ Γ.pk, private key $=$ Γ.sk) to be used for encryption and decryption. Classes and properties required for Σ and Γ suitable for designated confirmer signatures have been described and analyzed in [19, 20]. To give a specific example, Σ will be represented by a suitable *RSA* signature scheme and Γ by ElGamal.

The Full Domain Hash RSA [21] signature scheme Σ is given by the key pair (Σ.pk=(N, e), Σ.sk $= d$) where N is an RSA modulus and $ed \equiv 1 \bmod \phi(N)$. The keys

Table 1 Comparison with other work

Feature	[11]	[13]	[14]	[15]	[16]	Our method
Added security features in addition to the sequences	yes	no	no	n/a	no	yes
Source compression (via any cryptographic encoding, substitution cipher, or coding theory)	yes	yes	yes	n/a	n/a	yes
Explicit formulation and identification of broader security requirements and goals	no	no	no	n/a	no	yes
Adaptation and incorporation of novel cryptographic techniques to ensure that sensitive information remains con cealed during the verification process	no	no	no	no	no	yes

here are those used by the signer. With all computations in $\mathbf{Z}_N^*$, a valid signature σ on a message m is defined via $\sigma = H(m)^{\mathrm{d}}$, where H is a public hash function. The verification equation will make use of the following one-way function and image

$$f(x) = x^e \text{and } I = H(m). \tag{1}$$

Importantly, f is homomorphic as for all $x, y \in \mathbf{Z}_N^*$, $f(xy) = f(x)f(y)$.

For the encryption scheme Γ we use ElGamal's encyption [22]. It operates in a group $(\mathcal{G}, \cdot) =< g >$ of large enough order where computing discrete logarithms to base g is difficult. The confirmer's secret key is $\Gamma.\mathrm{sk} = x$ and the corresponding public key is $\Gamma.\mathrm{pk} = y = g^x$. To encrypt a message $m \in \mathcal{G}$, one chooses a random r and computes the ciphertext as the pair $\mathcal{E}(m) = (g^r, m \cdot y^r)$. To decrypt a ciphertext (K_1, K_2), one first obtains the session key $k = K_1^x = y^r$ and then computes m as $K_2 \cdot k^{-1}$.

Let $\circ$, the binary operation defined on $\mathcal{G} \times \mathcal{G}$, be the term-wise product $(a, b) \circ (c, d) = (ac, bd)$. Fundamental for the construction is the fact that ElGamal is homomorphic since [20],

$$\begin{aligned} \mathcal{E}(m) \circ \mathcal{E}(m') &= (g^r, m \cdot y^r) \circ (g^s, m' \cdot y^s) \\ &= (g^{r+s}, mm' \cdot y^{r+s}) = \mathcal{E}(mm'). \end{aligned} \tag{2}$$

The space of signatures produced by Σ must be the same as the space of messages encrypted by Γ. This can be done as follows: the signer chooses two sufficiently large primes p and q such that $p' = (p-1)/2$ and $q' = (q-1)/2$ are prime. The signer sets $N = pq$ and chooses $g \in \mathbf{Z}_N^*$ such that (with overwhelming probability) $Q_N = \{a^2 : a \in \mathbf{Z}_N^*\} \subseteq < g > \subseteq \mathbf{Z}_N^*$ and sets $\mathcal{G} = Q_N$. The signatures produced by Σ are mapped into $\mathcal{G}$ by squaring all the parameters (even the bases) before performing any modular operations with them. We also assume that the respective keys are verified with a certificate authority and the respective public parameters are publicly accessible. The symbol || will denote the operation which when applied to two strings m and z results in the 'usual' concatenation of the string m, and the string z.

The individual steps of our cryptographic protocol are described next. Let the given message m be the signature data to be signed. Throughout, if m is given in its binary representation (quartic representation as DNA), then after appropriate parsing $m \in \mathbf{Z}_N$ ($m \in \mathcal{G}$) is considered to be the representation of the integer m modulo N (in $\mathcal{G}$). The signer first generates a verifiable signature μ on m using the following steps:

(1) The signer chooses $r \overset{R}{\longleftarrow} \mathbf{Z}_{p'q'}$ and computes $z \longleftarrow g^r$ in $\mathcal{G}$.
(2) The signer uses his secret key $\Sigma.\mathrm{sk} = d$ of RSA-FDH to compute $\sigma = H(m||z)^{2d}$.
(3) σ is converted into DNA bases via our watermarking protocol below and hidden inside the GMO.
(4) The signer encrypts σ via ElGamal with the confirmer's public key $\Gamma.\mathrm{pk} = y = g^x$ and the random r, $\mathcal{E}(\sigma) = (K_1, K_2) = (g^r, \sigma \cdot y^r)$.
(5) $\mu = (K_1, K_2)$ is stored in an electronic database as the designated confirmer signature of m.

A candidate signature μ of m from a public database can only be validated by the TTP according to the following verification protocol.

(1) Given m and $\mu = (K_1, K_2)$, the confirmer computes σ as the decryption of the ElGamal ciphertext $(K_1 = z, K_2)$ using the secret key x.
(2) The confirmer verifies if σ is a valid RSA-FDH signature of $m||z$ by testing $\sigma^e = H(m||z)^2$ using the signer's public key e.
(3) The signature μ is accepted as valid if and only if this verification step passes.

Therefore, the algorithm runs as follows:

(1) Determine the number of occurrences N_i of each codon C_i in the host genome, see e.g. [23].
(2) Determine the number M_j of binary triplets B_j in the given binary sequence (with filling in of mock elements to yield a number of characters divisible by 3).
(3) Let $\mathcal{N} = \{A, C, G, T\}$ be the set of nucleotides. There are 24 ways in which these can be ordered. Let n_1, n_2 be the first two nucleotides, and n_3, n_4 the latter two in an arbitrary ordering.
(4) Associate with each triplets B_j of a given binary string a set of possible codons C_j, $B_1 \mapsto \{n_1n_1n_1,\ n_1n_1n_2,\ n_1n_2n_1, \ldots, n_2n_2n_2\}, \ldots B_8 \mapsto \{n_3n_3n_3,\ n_3n_3n_4,\ n_3n_4n_3, \ldots, n_4n_4n_4\}$, where the associated codon list excludes the ATG start codon [11].
(5) The number of times each text triplet is represented by each associated codon C_j is determined according to the following: For each B_j determine the number of occurrences N_j of each of the eight codons C_j that is associated to B_j via the above mapping. Spread out the N_j occurrences of each B_j according to match the individual numbers of occurrences of their associated codons C_j.

Results and discussion

The main difference between confirmer signatures and ordinary electronic signatures based on watermarking techniques is that confirmer signatures are not self-authenticating. Our algorithm is essentially based on the confirmer signature concept. In this case, the entity requiring proof of authentication (the "verifer") cannot check the ownership or the validity of a signature unless a

legitimate party confirms or disavows it [24, 25]. While the original signer (the originating product owner, developer or a delegate) can confirm the signature, it also may be verified by a semi-trusted third party (TTP) "confirmer" [26]. There are several advantages to limiting signature validation access to the confirmer and verifier. First, it protects the signer against coercion. Second, it protects the buyer if the signer becomes unavailable. Third, it validates the authentic signature without disclosing the actual signature sequence to an adversary who may even masquerade as a verifier. While the constant involvement of a TTP in the cryptographic protocol gives more power to the TTP, it also enables the TTP to obtain and disclose the signature in case of dispute and has additional important forensic implications. Following successful protocol implementation, the verifier and TTP have certifiable validation that the GMO contains a specific identifying signature although the verifier never acquires the exact signature [20].

Specifically, the confirmer and/or signer provides a ZK proof to demonstrate the validity of this signature to the verifier (Fig. 1). A valid signature is not accepted without the confirmation protocol, and a falsely alleged signature can only be repudiated via the denial protocol. In Figs. 1 and 2, the prover is either the confirmer of the signature who can undo encryption via ElGamal with the knowledge of the private key, or the signer who wishes to confirm the validity of signature μ. Thus, signature verification can be established by the signer without the involvement of the TTP. The TTP has the ability to undo ElGamal encryption and is the only party who can obtain the signature σ. Signature verification is therefore solely based on the encrypted signature $\mu = \mathcal{E}(\sigma)$, not the signature σ itself. In case of dispute the TTP can make σ public, convert it into nucleotides, and determine presence in the GMO.

Step 4 in Fig. 1 requires a protocol for proving that a given ciphertext under ElGamal decrypts to a given message M. The general building block is termed the proof of equality of two discrete logarithms [27, 28]. Adapted to our context, this protocol runs as depicted in Fig. 2. This protocol can be modified into a ZK proof of knowledge (ZKPOK) protocol by augmenting the Fig. 2 process by steps (2)–(4) of Fig. 1 [20].

It is important to note that a signature cannot be verified solely from identification of a unique sequence string. With our algorithm, as opposed to standard watermarking methods, a signature is not accepted as valid without the cooperation of the prover or delegate through the cryptographic confirmation protocol. Hence, even if an attacker detects a candidate signature, its validity is known only to the legitimate verifier who interacts with the prover in the protocol. Without the prover, no party can determine whether σ is a valid signature for m or not. Similarly, the specific denial protocol process ensures that a certain string cannot be denied by the original signer as an invalid signature. The protocol provides objective

Step	Prover		Verifier
(1)	Compute $I = H(m\|\|z)^2$		Compute $I = H(m\|\|z)^2$
(2)	Choose $s' \xleftarrow{R} \mathcal{G}$ Choose random $s \in \mathbb{Z}_{p'q'}$ Compute $\mathcal{E}(s') = (g^s, s' \cdot y^s)$, $t_1 = I \cdot (s')^e$, $t_2 = \mathcal{E}(s') \circ (K_1, K_2)$	$\xrightarrow{(t_1,t_2)}$	
(3)		$\xleftarrow{c}$	Choose $c \xleftarrow{R} \{0,1\}$
(4)	If $c = 0$:	$\xrightarrow{s,s'}$	Test if t_1, t_2 are computed as in (2)
	Otherwise:	$\xrightarrow{s' \cdot \sigma}$ ZKPOK that t_2 is encryption of $s' \cdot \sigma$ $\longleftrightarrow$	
(5)			Accept, if the ZKPOK is valid and, $(s' \cdot \sigma)^e = t_1$ in case of confirmation, $(s' \cdot \sigma)^e \neq t_1$ in case of denial.

Fig. 1 ZK proof of knowledge to verify authenticity. The prover and verifier are given the public input, an alleged signature $(K_1, K_2) = \mu$ with $z = K_1$, and the message (signature data) m. If $\mu = \boldsymbol{\mathcal{E}}(\sigma)$ is generated as above, then $t_1 = (s' \cdot \sigma)^e$ and $t_2 = (g^{s+r}, s'\sigma \cdot y^{s+r})$ where s is the randomness used in ElGamal to encrypt s', and r is that used to encrypt σ. In this case, the protocol allows the prover to confirm the signature in ZK. If μ is a falsely implied signature, the protocol allows the prover to deny the signature in ZK

Step	Prover		Verifier
(1)	Choose random r, t. Compute $K_1 = g^r, K_2 = My^r$, $a = y^t, b = g^t$.		
		(K_1, K_2, a, b) →	
(2)			Choose random challenge $c \in \{0,1\}^*$
		← c	
(3)		$s = t + cr \bmod p'q'$ →	
(4)			Compute $W = K_2/M$. Accept the proof as valid if $y^s = aW^c$ and $g^s = bK_1^c$.

Fig. 2 Proof that (K_1, K_2) is the encryption of the given message M under ElGamal. If the prover can successfully answer two distinct challenges c_1, c_2 with two acceptable answers s_1, s_2 then the verification step results in $y^{s_1-s_2} = W^{c_1-c_2}$ and $g^{s_1-s_2} = K_1^{c_1-c_2}$ (see [27]). Thus, if $c_1 - c_2 < p'q'$ a value r exists such that $r = \log_y W = \log_g K_1 = (s_1 - s_2)/(c_1 - c_2) \bmod p'q'$. Consequently, $W = y^r$, $K_1 = g^r$, and $K_2 = My^r$. This proves that (K_1, K_2) is indeed an encryption of the message M under ElGamal that can be translated into a ZKPOK according to [20]

certifiable authentication of ownership as the signature is retrievable and verifiable by designated parties, e.g. the TTP. The TTP may disclose the signature for verification by other parties although normally, the company's signature data remain completely hidden. The ZK property ensures that no one has access to the signer's secret key or the signature. Even if the buyer or a masquerading adversary analyzes via genome sequencing individual GMO's in a population, the secret key or the signature that allows them to impersonate the verifier cannot be discovered.

The protocol ensures that the signature string is indistinguishable in the electronic signature space as represented by integers [20, 29]. In conjunction with the signature protocol we developed a watermarking algorithm that is designed to provide signature invisibility. The protocol consists of converting the cryptographic signature σ into the DNA alphabet such that it is indistinguishable from the endogenous DNA after insertion in the genome. The algorithm effectively camouflages the required authentication and/or tracking data to ensure that an adversary cannot identify the signature as a security or watermarking feature. The process also is reversible. From the nucleotide sequence, the signature can be translated to cryptographic code. The algorithm encodes binary triplets based on the frequency of each codon as determined by the codon bias of the host. Codon bias refers to each organism's inherent preferences of certain triplet nucleotides for translation into corresponding amino acids. Our approach is solely DNA based and does not require transcription or translation of sequences. Alignment of the message with the host codon bias is designed to better hide the message in the genomic background. Each binary bit is assigned to a choice of two specific nucleotides $n_i \in \{A, C, G, T\}$, i.e. $0 \mapsto n_1$ or n_2, $1 \mapsto n_3$ or n_4 to mirror the codon frequencies of the host with the frequencies of the binary text triplets. Figure 3 shows the correspondence between the binary text triplets B_i and DNA codons C_i for the specific example where $n_1 = A, n_2 = C, n_3 = G, n_4 = T$. Each of the text triplets is distributed over the associated codon triplets so that the resulting representation resembles the codon bias of the host genome. Further improvement can be made by renaming and reordering, as the choice of the n_i is arbitrary. Renaming the codons by matching the obtained string with the host frequency distribution results in a correlation of the obtained with the host frequency distribution of typically 0.98 or more.

Therefore, each of the binary triplets is represented in terms of their associated codons according to the number of occurrences. Figure 4 shows the correlation to the overall *E. coli* genome codon frequencies that is obtained when an arbitrarily chosen binary signature of length 1000 is represented in terms of DNA nucleotides as determined by our algorithm.

Following design and construction of the entire nucleotide array encompassing the message and algorithm components, there are numerous approaches to incorporate the construct stably in the target organism using standard biotechnological tools for prokaryotes and eukaryotes.

Conclusions

Our protocol is provably secure in terms of standard cryptanalytic tools, and integrates advanced electronic signature methods with a new watermarking or data-embedding technique, yielding a highly secure and authentication-based product. Importantly, the authentic signature is indistinguishable from random elements in the signature space and the authentication string can be confirmed or denied without disclosing the actual

text triplet B_i	set $\{n_1n_1n_1,\ n_1n_1n_2,\ n_1n_2n_1, \ldots, n_4n_4n_4\}$ of associated codons C_i
$B_1 = 000$	{AAA, AAC, ACA, ACC, CAA, CAC, CCA, CCC}
$B_2 = 001$	{AAG, AAT, ACG, ACT, CAG, CAT, CCG, CCT}
$B_3 = 010$	{AGA, AGC, ATA, ATC, CGA, CGC, CTA, CTC}
$B_4 = 100$	{GAA, GAC, GCA, GCC, TAA, TAC, TCA, TCC}
$B_5 = 011$	{AGG, AGT, ATG, ATT, CGG, CGT, CTG, CTT}
$B_6 = 101$	{GAG, GAT, GCG, GCT, TAG, TAT, TCG, TCT}
$B_7 = 110$	{GGA, GGC, GTA, GTC, TGA, TGC, TTA, TTC}
$B_8 = 111$	{GGG, GGT, GTG, GTT, TGG, TGT, TTG, TTT}

Fig. 3 Correspondence between the binary text triplets B_i and DNA codons C_i for the specific example where $n_1 = A, n_2 = C, n_3 = G, n_4 = T$. Each of the text triplets is distributed over the associated codon triplets so that the resulting representation resembles the codon bias of the host genome. To demonstrate the watermarking protocol, assume there are 44 occurrences of 000 in the binary text and that the codon frequency values as determined from the entire codon frequency distribution, are: *AAA*, 3.3 %, *AAC*, 2.1 %, *ACA*, 0.8 %, *ACC*, 2.3 %, *CAA*, 1.5 %, *CAC*, 0.9 %, *CCA*, 0.8 %, *CCC*, 0.6 %, covering a total of 12.5 % of the total codon distribution. Among the codons assigned to 000, there are $100 \cdot 3.3/12.5 = 26.6$ % for *AAA*, 17.1 % for *AAC* etc. Consequently, we assign $(26.6 \cdot 44)/100 \sim 12$ occurrences of 000 to *AAA*, 8 to *AAC*, 3 to *ACA*, 8 to *ACC*, 5 to *CAA*, 3 to *CAC*, 3 to *CCA*, and 2 to *CCC*, covering the total 44 occurrences of 000

signature. The signature data are not strictly limited in sequence size and may include, but are not limited to, security details, the product production and distribution chain and company licensing details. The resultant data are used as input to establish the security signature in binary according to an algorithm that yields a corresponding string with a nucleotide distribution that closely reflects the natural codon bias of the given host genome. The resulting sequence construct may be inserted into the GMO genome using established genetic engineering technologies such that it is stably inherited through generations. Further increasing the security level can be accomplished by inserting the authentic signature into a subset of the GMO population with the remaining population containing imitation signatures. Authentic signature clones may be identified via PCR by the legitimate owner of the signature or by a designated judge. Importantly, the signature key allowing decoding and authentication is not revealed during this process, thus allowing continued utilization of the key. This provides an increased level of security against whole-genome sequencing and alignment that might increase the probability of identifying a security signature with other standard watermarking approaches.

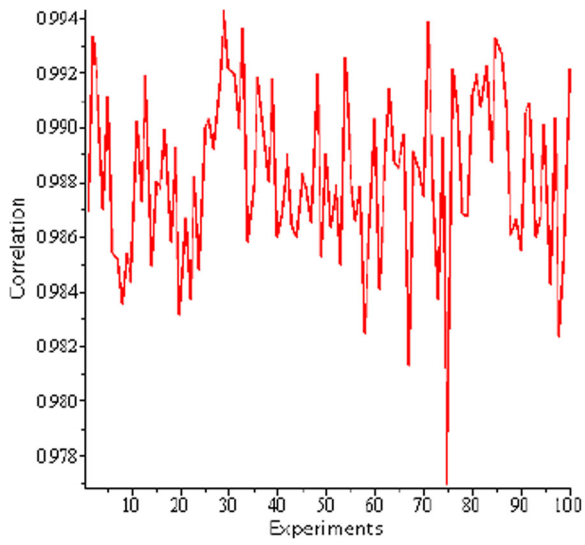

Fig. 4 Representation of the correlation between codon frequency distribution produced via our watermarking algorithm and the initial codon distribution of the host genome (y-axis). The watermarking algorithm generates a signature indistinguishable from the rest of the genome. The x-axis here is 100 randomly generated binary signatures of length 1000 mapped to the codon frequency distribution of the individual codons in E. coli

Acknowledgments
The authors thank the anonymous referees for their careful reading of this manuscript and their extensive comments. The paper has improved with these changes.

Funding
None.

Availability of data and materials
The data in this study were computer generated. The codes were implemented in Maple, and have been described extensively in the manuscript. The algorithm represents a theoretical contribution to the literature; the implementation itself is not necessary for public distribution.

Authors' contributions
SM is responsible for developing the algorithm and the project. DR envisioned the project, DR and FJ oversaw the project and all authors cowrote the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Author details
[1]Department of Molecular Biology, University of Wyoming, 1000 E. University Ave, 82071 Laramie, WY, USA. [2]Department of Mathematics, University of Wyoming, 1000 E. University Ave, 82071 Laramie, WY, USA.

Received: 25 November 2015 Accepted: 9 September 2016
Published online: 21 September 2016

References

1. Cachin C. An information-theoretic model for steganography. In: Information Hiding. New York: Springer Publishing Co.; 1998. p. 306–18.
2. Anderson RJ, Petitcolas FA. On the limits of steganography. IEEE J Sel Areas Commun. 1998;16(4):474–81.
3. Von Ahn L, Hopper NJ. Public-key steganography. In: Advances in Cryptology-EUROCRYPT 2004. New York: Springer Publishing Co.; 2004. p. 323–41.
4. Backes M, Cachin C. Public-key steganography with active attacks. In: Theory of Cryptography, Volume 3378 of the Series Lecture Notes in Computer Science. New York: Springer Publishing Co.; 2005. p. 210–26.
5. Clelland CT, Risca V, Bancroft C. Hiding messages in DNA microdots. Nature. 1999;399(6736):533–4.
6. Leier A, Richter C, Banzhaf W, Rauhe H. Cryptography with DNA binary strands. BioSystems. 2000;57(1):13–22.
7. Shimanovsky B, Feng J, Potkonjak M. Hiding data in DNA. In: Information Hiding. New York: Springer Publishing Co.; 2003. p. 373–86.
8. Wong PC, Wong K-k, Foote H. Organic data memory using the DNA approach. Commun ACM. 2003;46(1):95–8.
9. Arita M, Ohashi Y. Secret signatures inside genomic DNA. Biotechnol Prog. 2004;20(5):1605–7.
10. Jupiter DC, Ficht TA, Samuel J, Qin QM, de Figueiredo P. DNA watermarking of infectious agents: progress and prospects. PLoS Pathog. 2010;6(6):1000950.
11. Haughton D, Balado F. Biocode: Two biologically compatible algorithms for embedding data in non-coding and coding regions of DNA. BMC Bioinforma. 2013;14:121.
12. Müller S, Jafari F, Roth D. Improving the dependability and precision of artificial DNA for information-theoretic purposes. Submitted. Technical Report. 2015. doi:10.13140/RG.2.1.1215.8325.
13. Heider D, Barnekow A. DNA-based watermarks using the DNA-Crypt algorithm. BMC Bioinforma. 2007;8:176.
14. Heider D, Barnekow A. DNA watermarks: A proof of concept. BMC Mol Biol. 2008;9:40.
15. Yachie N, Ohashi Y, Tomita M. Stabilizing synthetic data in the DNA of living organisms. Syst Synth Biol. doi:10.1007/s11693-008-9020-5.
16. Gibson DG, et al. Creation of a bacterial cell controlled by a chemically synthesized genome. Science. 2010;329:52.
17. Balado F, Haughton D. Gene tagging and the data hiding rate, ISSC. 2012;52–56.
18. Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof systems. SIAM J Comput. 1989;18(1):186–208.
19. Camenisch J, Michels M. Confirmer signature schemes secure against adaptive adversaries. In: Advances in Cryptology-EUROCRYPT 2000. New York: Springer Publishing Co.; 2000. p. 243–58.
20. El Aimani L. On generic constructions of designated confirmer signatures. In: Progress in Cryptology - INDOCRYPT 2009. New York: Springer Publishing Co.; 2009. p. 343–62.
21. Bellare M, Rogaway P. The exact security of digital signatures-how to sign with RSA and Rabin. In: Advances in Cryptology - Eurocrypt'96. New York: Springer Publishing Co.; 1996. p. 399–416.
22. ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory. 1985;31(4):469–72.
23. Codon Usage Database. http://www.kazusa.or.jp/codon/.
24. Chaum D, Van Antwerpen H. Undeniable signatures. In: Advances in Cryptology-CRYPTO'89 Proceedings. New York: Springer Publishing Co.; 1990. p. 212–6.
25. Chaum D. Zero-knowledge undeniable signatures. In: Eurocrypt. New York: Springer Publishing Co.; 1990. p. 458–64.
26. Chaum D. Designated confirmer signatures. In: Advances in Cryptology-EUROCRYPT'94. New York: Springer Publishing Co.; 1995. p. 86–91.
27. Chaum D, Pedersen TP. Wallet databases with observers. In: Advances in Cryptology-CRYPTO'92. New York: Springer Publishing Co.; 1993. p. 89–105.
28. Ateniese G. Verifiable encryption of digital signatures and applications. ACM Trans Inf Syst Secur (TISSEC). 2004;7(1):1–20.
29. Galbraith SD, Mao W. Invisibility and anonymity of undeniable and confirmer signatures. In: Topics in Cryptology - CT-RSA 2003, LNCS 2612. New York: Springer Publishing Co.; 2003. p. 80–97.