# Breaking BASS

Simon-Philipp Merz[1], Kenneth G. Paterson[1], and Àlex Rodríguez García[2]

[1] Applied Cryptography Group, ETH Zurich
[2] Universitat Politècnica de Catalunya

**Abstract.** We provide several attacks on the BASS signature scheme introduced by Grigoriev, Ilmer, Ovchinnikov and Shpilrain in 2023. We lay out a trivial forgery attack which generates signatures passing the scheme's probabilistic signature verification with high probability. Generating these forgeries is faster than generating signatures honestly. Moreover, we describe a key-only attack which allows us to recover an equivalent private key from a signer's public key. The time complexity of this recovery is asymptotically the same as that of signing messages.

## 1 Introduction

In this paper, we present multiple attacks against the Boolean automorphisms signature scheme (BASS) proposed by Grigoriev, Ilmer, Ovchinnikov and Shpilrain in 2023 [1].

First, we show that the scheme's probabilistic signature verification allows us to forge signatures that will be likely to pass verification. Producing these forgeries is more efficient than honestly generating signatures using the private key. According to our experiments, the forgeries produced this way pass the verification algorithm of BASS on average 86.2% of the time for the proposed parameters.

Second, we provide a key-only attack which recovers a key equivalent to the private key from the public key. This allows us to forge signatures that pass verification with the same probability as legitimately generated signatures. This attack is practical for the suggested parameters of the signature algorithm and it runs with the same complexity as the signing algorithm asymptotically.

Finally, we briefly discuss how to further improve our attack by exploiting how secret keys are chosen in BASS.

Along the way, we provide a different characterisation of the space of the Boolean automorphisms used as private keys in BASS. From our description it is apparent that the private keys are far from being unique. Further, we discuss the probability of the BASS verification algorithm rejecting valid signatures. Our analysis and experimental results diverge significantly from the claims made by the authors of BASS [1].

[2] Authors listed in alphabetical order: see https://www.ams.org/profession/leaders/CultureStatement04.pdf. The research was carried out while the third author visited ETH Zurich on an SSRF scholarship.

All algorithms described in this paper are implemented and were used to verify our results experimentally. The code is available at [2].

*Outline.* In Section 2 we introduce some necessary notation as well as the notion of two bases that will be crucial to describe this paper's contributions. In Section 3 we recall the BASS signature scheme and we provide our own explanation of why the probabilistic verification of the scheme is usually correct. We further explain why it seems to have more false negatives than claimed in [1]. Section 4 characterises (the large number) of equivalent private keys in the BASS signature scheme. In Section 5 we give a probabilistic attack which forges signatures that will likely be accepted. Generating forgeries takes less time than it would take to honestly sign a message. Finally, in Section 6, we show how to recover an equivalent private key from a public key in BASS. Using this, we can forge signatures that will be accepted with the same probability as legitimately generated signatures.

## 2 Notation

We introduce the following notation to describe the BASS signature scheme and our attacks. Let $K_n = \mathbb{Z}[x_1, \ldots, x_n]$ and $I$ the ideal in $K_n$ generated by the polynomials $x_i^2 - x_i$ for $i \in [n]$. The ring $B_n := K_n / I$ is called the "booleanization" of $K_n$. We denote the automorphisms of $B_n$ by $\mathrm{Aut}(B_n)$. Let $c = c_1 c_2 \ldots c_n \in \{0,1\}^n$ be an $n$-tuple of bits, i.e. $c_i \in \{0,1\}$. Further, we define

$$b_c := \prod_{i=1}^{n} x_i^{c_i} \in B_n \qquad \mathbb{B} := \{b_c \mid c \in \{0,1\}^n\},$$

$$e_c := \prod_{i=1}^{n} x_i^{c_i}(1-x_i)^{1-c_i} \in B_n \qquad \mathbb{E} := \{e_c \mid c \in \{0,1\}^n\}.$$

We refer to $\mathbb{B}$ as the *standard basis* of $B_n$ and $\mathbb{E}$ as the *orthogonal basis* of $B_n$. Given $p \in B_n$, we denote by $p(c) = p(c_1, c_2, \ldots, c_n) = p|_{x_i = c_i\ \forall i \in [n]}$ the evaluation of $p$ at a Boolean $n$-tuple $c \in \{0,1\}^n$. Partial evaluations are written as $p|_{x_i = b}$, where $b \in \{0,1\}$.

When providing implementation details, we denote a polynomial $p$ represented in the standard basis by $[p]_b$ and in the orthogonal basis by $[p]_e$. In the first case, polynomials are represented as a list of monomials whose size depends on the number of monomials. In the second case, polynomials are represented by a list of $2^n$ binary integer coefficients, where the $i$-th element corresponds to the coefficient of $e_i$. In all implementations we represent automorphisms by their action on the orthogonal basis, leaving no ambiguity in their representation.

## 3 The BASS signature scheme

In this section, we briefly recall the BASS signature scheme with the parameters suggested in [1].

**Key generation.** A secret automorphism $\phi \in \mathrm{Aut}(B_{31})$ is sampled as the private key of the signer. Further, three sparse polynomials $P_1, P_2, P_3 \in B_{31}$ are chosen at random from a publicly known distribution. The public key consists of the polynomials $P_1, P_2, P_3, \phi(P_1), \phi(P_2), \phi(P_3) \in B_{31}$. Using the recommended parameters, all three $P_i$ are the sum of three monomial terms with coefficients in $\{-1, 1\}$.

**Signature generation.** To sign a message $\mathfrak{m}$, it is hashed onto a 256-bit string (using SHA3-256) which is encoded as a polynomial $Q \in B_{32}$ according to a publicly known map. Then, the secret automorphism $\phi$ is extended to $\phi_{\mathrm{ext}} \in \mathrm{Aut}(B_{32})$ such that $\phi_{\mathrm{ext}}(x_i) = \phi(x_i),\ \forall i \in [31]$ and a randomly defined image for $\phi_{\mathrm{ext}}(x_{32})$ that is compatible with $\phi_{\mathrm{ext}}$ being an automorphism is selected. The signature for $\mathfrak{m}$ is $\sigma := \phi_{\mathrm{ext}}(Q) \in B_{32}$.

**Verification.** When verifying a signature, first a random polynomial $u \in K_4$ is sampled. Given the message-signature pair $(\mathfrak{m}, \sigma)$, $\mathfrak{m}$ is mapped to the polynomial $Q \in B_{32}$ following the same procedure as in the signature generation. Let

$$R := u(P_1,\, P_2,\, P_3,\, Q),\ S := u(\phi(P_1),\, \phi(P_2),\, \phi(P_3),\, \sigma) \in B_{32}.$$

The verifier samples with repetition 3000 Boolean tuples in $\{0,1\}^{32}$ and evaluates both $R$ and $S$ at these tuples and counts the tuples that evaluate to a positive value. If the number of positive values appearing for $R$ and $S$ differ by no more than 3% of the total number of samples, the signature is accepted as valid and rejected otherwise. Note, this is a probabilistic signature verification algorithm.

*Remark 3.1.* The description of the signature verification in BASS is not consistent throughout [1]. In most places, the authors claim that the proportions of positive values of $R$ and $S$ are compared as mentioned above [1, Sect. 2-7]. However, according to the abstract and the proof of concept implementation the number of zeroes in $R$ and $S$ are compared. Our attacks later apply to both verification procedures, but for consistency we will adhere to the description given above.

### 3.1 Correctness of the verification algorithm

We provide in this subsection our own explanation for the correctness of the signature scheme, i.e. why we expect the scheme to verify legitimately generated signatures. We typically express polynomials in $B_n$ using the standard basis. However, the orthogonal basis is more useful for understanding the behaviour of automorphisms. The following lemma summarises the key properties of the orthogonal basis that we will use in the following. Its proof is located in Appendix A.

**Lemma 3.1** *Let* $\mathbb{E} := \{e_c \mid c \in \{0,1\}^n\}$ *with* $e_c := \prod_{i=1}^{n} x_i^{c_i}(1-x_i)^{1-c_i} \in B_n$ *denote the "orthogonal basis".*

*(i)* *The orthogonal basis is indeed a basis, i.e. any polynomial in $B_n$ can be uniquely expressed as a linear combination of elements in $\mathbb{E}$.*

*(ii)* *For $c, d \in \{0,1\}^n$ with $c \neq d$, we have $e_c(d) = 0$ and $e_c \cdot e_d = 0$. Moreover, $e_c(c) = 1$ and $e_c \cdot e_c = e_c$.*

*(iii)* *The orthogonal basis is a Lagrange basis, i.e. any polynomial $A \in B_n$ can be expressed in terms of $\mathbb{E}$ as*

$$A = \sum_{c \in \{0,1\}^n} A(c) e_c.$$

*(iv)* *For each $\phi \in \mathrm{Aut}(B_n)$ there exists a permutation $\pi \in \Sigma_{\{0,1\}^n}$ such that $\phi(e_c) = e_{\pi(c)}$ for every $c \in \{0,1\}^n$.*

*(v)* *For each $\pi \in \Sigma_{\{0,1\}^n}$ there exists an automorphism $\phi \in \mathrm{Aut}(B_n)$ such that $\phi(e_c) = e_{\pi(c)}$ for every $c \in \{0,1\}^n$.*

Lemma 3.1 shows that any automorphism can be viewed as a permutation on the variables of $B_n$. We use this result to argue why honestly generated signatures are expected to pass verification.

Given any polynomial $A \in B_n$ and any automorphism $\phi \in \mathrm{Aut}(B_n)$, we can write

$$\begin{aligned} A &= \sum_{c \in \{0,1\}^n} A(c) e_c \\ \phi(A) &= \sum_{c \in \{0,1\}^n} A(c) \phi(e_c) = \sum_{c \in \{0,1\}^n} A(c) e_{\pi(c)} = \sum_{c \in \{0,1\}^n} A(\pi^{-1}(c)) e_c \end{aligned} \tag{1}$$

for some permutation $\pi$. Then, $\phi(A)(c) = A(\pi^{-1}(c))$. Here we have used *(ii)*, *(iii)* and *(iv)* from Lemma 3.1 and linearity of automorphisms. If $\sigma$ is an honestly generated signature, then $\sigma = \phi_{\mathrm{ext}}(Q)$. Using linearity of automorphisms, we have

$$\begin{aligned} R &= u(P_1, P_2, P_3, Q) \\ S &= u\big(\phi_{\mathrm{ext}}(P_1), \phi_{\mathrm{ext}}(P_2), \phi_{\mathrm{ext}}(P_3), \phi_{\mathrm{ext}}(Q)\big) \\ &= \phi_{\mathrm{ext}}\big(u(P_1, P_2, P_3, Q)\big) = \phi_{\mathrm{ext}}(R). \end{aligned}$$

Using Eq. (1), we obtain: $R(c) = S(\pi^{-1}(c))$ for some permutation $\pi$. Using this relationship, it is easy to see that if we evaluate the polynomials $R$ and $S$ on the set of all Boolean tuples the results have to be the same up to a permutation. In particular, the number of positive images have to be the same.

**Probability of false negatives.** Let $r$ be the proportion of positive images when evaluating $R$ or $S$ on all Boolean tuples. When evaluating the polynomials on a sample of 3000 values, the number of positive images follows a binomial distribution $\mathrm{Bin}(3000, r)$ as the verification algorithm samples without replacement.

When running the signature verification on an honestly generated signature, the probability that this signature is falsely rejected is the probability that two random variables $X, Y \sim \mathrm{Bin}(3000, r)$ satisfy $|X - Y| > 90$, i.e. that the proportions differ by more than 3% of the samples. When approximating $X, Y$ as normal distributions $N(3000r,\, 3000r(1-r))$, then $X - Y \sim N(0,\, 6000r(1-r))$. Using the fact that $r(1-r) \leq 1/4$ and looking at the normal probability density function table, we conclude

$$\Pr(|X - Y| \leq 90) \leq \Pr(-2.32\sigma \leq X - Y \leq 2.32\sigma) \approx 0.98,$$

where $\sigma$ denotes the standard deviation. Hence the probability of rejecting an honestly generated signature is at most 2%.

Note that this bound is tight if the proportion of positive images among all the evaluations on Boolean tuples $r$ is close to 0.5 which can be observed in practice. We ran the probabilistic signature verification on a fixed message-signature pair 100.000 times. The signature was falsely rejected in roughly 1.32% of the trials. The example can be found at [2].

This contrasts the claim by the authors of BASS that false negatives would only occur with a tiny probability of $2^{-33}$.

## 4 Space of equivalent private keys

In this section, we characterise automorphisms which are equivalent to a signer's private key in the BASS signature scheme, i.e. automorphisms that allow us to compute signatures which will pass verification with the same probability as signatures generated with the private key of the signer, independent of the message signed.

### 4.1 A sufficient condition for equivalent private keys

**Lemma 4.1** *Let $\phi \in \mathrm{Aut}(B_{32})$ be a signer's private key, let $P_1, P_2, P_3$ be the polynomials contained in the public key and let $Q \in B_{32}$ be the polynomial corresponding to a message $\mathfrak{m}$. For any $\psi \in \mathrm{Aut}(B_{32})$ satisfying*

$$\psi(P_i) = \phi(P_i),\ \forall i \in \{1, 2, 3\}, \tag{2}$$

*the polynomial $\psi(Q)$ will be accepted as a valid signature with the same probability as an honestly generated signature $\phi(Q)$.*

*Proof.* Using the same notation as in Section 3, note that

$$\begin{aligned} S &= u(\phi(P_1), \phi(P_2), \phi(P_3), \psi(Q)) = u(\psi(P_1), \psi(P_2), \psi(P_3), \psi(Q)) \\ &= \psi(u(P_1, P_2, P_3, Q)) = \psi(R) \end{aligned}$$

The verification only checks (probabilistically) the equality of the proportion of positive images of $S$ and $R$. Since $\psi$ is an automorphism the proportion of

positive values are the same for both polynomials, following the same argument as in Section 3 for honestly generated signatures. Hence, the signature $\psi(Q)$ will be accepted by the verifier with the same probability as an honestly generated signature. □

As such, Lemma 4.1 provides a sufficient condition on whether an automorphism $\psi \in \mathrm{Aut}(B_{32})$ is equivalent to a signer's private key for the purpose of producing arbitrary valid message-signature pairs.

Recall from the key generation in Section 3, that all three polynomials $P_i$ in the public key are the sum of three monomial terms with coefficients in $\{-1, 1\}$. Let $a_i$ denote the number of coefficients in $P_i$ equal to $-1$. As each term of $P_i$ only takes values in $\{0, 1\}$ when evaluated at a Boolean tuple, the polynomial $P_i - a_i$ takes values in the set $\{0, 1, 2, 3\}$ for each $i \in \{1, 2, 3\}$. Consider the polynomial $P$ defined as

$$P := (P_1 - a_1) + 4 \cdot (P_2 - a_2) + 16 \cdot (P_3 - a_3). \tag{3}$$

Note that the polynomial

$$\phi(P) = (\phi(P_1) - a_1) + 4 \cdot (\phi(P_2) - a_2) + 16 \cdot (\phi(P_3) - a_3)$$

can also be computed from the public key.

Clearly, any $\psi \in \mathrm{Aut}(B_{32})$ satisfying $\psi(P) = \phi(P)$ also satisfies the sufficient condition from Lemma 4.1, as we have only rewritten the condition in base 4.

Thus, to recover a key that is equivalent to a private key of a signer it is sufficient to solve the following problem (in BASS with $n = 32$).

*Problem 4.1.* Let $\phi \in \mathrm{Aut}(B_n)$ be an unknown automorphism. Given two polynomials $P, \phi(P) \in B_n$, find $\psi \in \mathrm{Aut}(B_n)$ such that $\psi(P) = \phi(P)$.

Our observation about equivalent private keys raises the question of how many solutions there are for Problem 4.1.

### 4.2 Solutions to Problem 4.1 as permutations

In this section we characterise the set of solutions to Problem 4.1 as a set of permutations with a specific property described by the following theorem. This characterisation will allow us to compute the number of solutions to Problem 4.1.

**Theorem 4.2.** *Let $A, B \in B_n$ be two polynomials and let*

$$V := \{v \in \mathbb{Z} \mid \exists c \in \{0,1\}^n \text{ such that } A(c) = v \text{ or } B(c) = v\}.$$

*We write $m = |V|$ and $V = \{v_i \mid i \in \{0, 1, \ldots, m-1\}\}$. For each $i \in \{0, 1, \ldots, m-1\}$, we further define*

$$S_i := \{c \in \{0,1\}^n \mid A(c) = v_i\} \text{ and } T_i := \{c \in \{0,1\}^n \mid B(c) = v_i\}.$$

*Then, there exists an automorphism $\psi \in \mathrm{Aut}(B_n)$ such that $\psi(A) = B$ if and only if $|S_i| = |T_i|$, for all $i \in \{0, 1, \ldots, m-1\}$.*

*Proof.* Using Lemma 3.1 *(iii)* and grouping terms based on the value of $A(c)$ and $B(c)$ respectively, we write

$$A = \sum_{i=0}^{m-1} v_i \left( \sum_{\substack{c \in \{0,1\}^n \\ A(c)=v_i}} e_c \right) = \sum_{i=0}^{m-1} v_i \left( \sum_{c \in S_i} e_c \right)$$

$$B = \sum_{i=0}^{m-1} v_i \left( \sum_{\substack{c \in \{0,1\}^n \\ B(c)=v_i}} e_c \right) = \sum_{i=0}^{m-1} v_i \left( \sum_{c \in T_i} e_c \right).$$

We first prove that if there exists $\psi \in \mathrm{Aut}(B_n)$ such that $\psi(A) = B$, then $|S_i| = |T_i|$ for all $i \in \{0, 1, \ldots, m-1\}$. Note that

$$\psi(A) = \sum_{i=0}^{m-1} v_i \left( \sum_{c \in S_i} e_{\tau(c)} \right) = \sum_{i=0}^{m-1} v_i \left( \sum_{c \in \tau(S_i)} e_c \right)$$

for some unknown permutation $\tau$. Using Lemma 3.1 *(iii)*, we know that the coefficients of the polynomials $\psi(A)$ and $B$ in the orthogonal basis must coincide. In particular, we have $\tau(S_i) = T_i$ and thus both sets must have the same cardinality.

We prove the reverse direction by construction. Suppose $|S_i| = |T_i|$ for all $i \in \{0, 1, \ldots, m-1\}$. Then we can define a bijection between elements of $S_i$ and $T_i$. As the sets $S_i$ are pairwise disjoint and so are the $T_i$, these bijections define a permutation $\tau$ over the set of Boolean tuples. Using Lemma 3.1 *(v)*, we know that $\tau$ defines an automorphism $\psi \in \mathrm{Aut}(B_{32})$. As

$$\psi(A) = \sum_{i=0}^{m-1} v_i \left( \sum_{c \in S_i} e_{\tau(c)} \right) = \sum_{i=0}^{m-1} v_i \left( \sum_{c \in \tau(S_i)} e_c \right) = \sum_{i=0}^{m-1} v_i \left( \sum_{c \in T_i} e_c \right) = B,$$

we obtain the desired equality. □

*Remark 4.3.* We can apply Theorem 4.2 to the polynomials $P$ and $\phi(P)$ defined in Eq. (3) and consider the respective sets $S_i$ and $T_i$. The space of solutions $\psi$ to Problem 4.1 corresponds to all the permutations $\tau$ such that $\tau(S_i) = T_i$, $\forall i \in \{0, 1, \ldots, m-1\}$. Note that any such permutation induces a solution to Problem 4.1. Thus, we can count the exact number of solutions, which is: $\prod_{i=0}^{m-1} |S_i|!$. For the proposed parameters of BASS and the definition of $P$, we know that $P$ takes values only in $\{0, 1, \ldots, 63\}$, i.e. $m \leq 64$. The value is minimised for larger $m$ and if each set $S_i$ has an equal number of elements. Therefore, the number of automorphisms $\psi \in \mathrm{Aut}(B_{32})$ such that $\psi(P) = \phi(P)$, i.e. the number of equivalent keys for the BASS signature scheme, is considerable – at least $(2^{32}/64)!^{64} \approx 10^{10^{10.5}}$. This contradicts the statement of the authors of BASS that "the secret key does not have to be unique, although most of the time it is" [1, Sect. 7].

## 5 Signing with trivial signatures

We have seen that there are many automorphisms which solve Problem 4.1. When evaluating any such automorphism on $Q$, the encoded hash of a message, we obtain a polynomial that will be accepted during the signature verification with the same probability as a legitimately generated signature. In Section 6 we will describe how to compute such equivalent keys. Another natural question is:

*Given a polynomial $\sigma \in B_n$ and a message corresponding to the hashed encoding $Q \in B_n$, how likely will the probabilistic signature verification accept $\sigma$?*

Note that a polynomial will likely be accepted whenever it is "sufficiently close" to a polynomial obtained by evaluating an automorphism $\psi$ from the set of equivalent private keys on $Q$.

In this section, we provide a lower bound for the above question. Estimating the probability of polynomials passing the signature verification will allow us to describe how to forge signatures that are likely to be accepted in less time than it would take to generate a legitimate signature using the private key.

More precisely, we will argue why just outputting $Q$, the encoded hash of a message, as a forgery, may pass signature verification. Then, we provide experimental evidence that such forgeries will indeed be accepted on average 86.2% of the time for randomly chosen pairs of messages and public keys.

Further, we observe experimentally that the trivial zero-polynomial will also pass verification on average 11.2% of the time for randomly chosen messages.

For a polynomial $\sigma$ to pass verification with high probability, it needs to be close to $\psi(Q)$ for one of the automorphisms $\psi$ solving Problem 4.1, i.e. we want both $\psi(Q)$ and $\sigma$ to evaluate to a positive value simultaneously on as many Boolean tuples as possible. For a fixed $Q$ and $\sigma$, a probability estimate for the acceptance can be approximated by determening the distance to a good choice of automorphism $\psi$ evaluated on $Q$.

The following example illustrates how to construct such an automorphism $\psi$ for the purpose of estimating the probability of a given polynomial $\sigma$ passing signature verification.

*Example 5.1.* For a small example with $n = 3$, suppose the polynomials $P$, $\phi(P) \in B_3$ computed from the public key are

$$[P]_e = [0, 1, 0, 2, 1, 0, 0, 1]$$
$$[\phi(P)]_e = [0, 0, 2, 1, 0, 1, 1, 0].$$

From Theorem 4.2, we know that finding an automorphism which maps $P$ to $\phi(P)$ is equivalent to finding a permutation that maps the first vector of coefficients to the second one. For example, we may consider the permutation $\tau_1 = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 4 & 2 & 3 & 6 & 5 & 8 & 7 \end{smallmatrix}\right)$, which induces an automorphism $\psi_1$. Suppose $Q$ is the encoding of the hash of a message $\mathfrak{m}$ and assume we want to sign the message with

the polynomial $\sigma$. These polynomials may look like

$$[Q]_e = [-1, 2, 2, 0, -1, 1, 1, 0]$$
$$[\sigma]_e = [1, -1, 0, 0, 2, -1, 1, 0].$$

We know that $\psi_1(Q)$ is a signature which will pass verification with the same probability as an honestly generated signature. We would have

$$[\psi_1(Q)]_e = [-1, 2, 0, 2, 1, -1, 0, 1].$$

Given that $\sigma$ and $\psi_1(Q)$ are represented in the orthogonal basis, we can compare the vectors componentwise to see that there are exactly two Boolean tuples where both polynomials evaluate to the same value. However, we can try to find an automorphism $\psi_2$ such that $\psi_2(Q)$ is a valid signature, and also "closer" to $\sigma$ in the sense that the evaluation coincides on more Boolean tuples. Let $\tau_2$ be the permutation corresponding to $\psi_2$. As $\tau_2$ has to map the vector of coefficients of $P$ to the vector of coefficients of $\phi(P)$, we have restrictions such as $\tau_2(1) \in \{1, 2, 5, 8\}$. If possible, we choose among the permissible images such that $\psi_2(Q)$ and $\sigma$ agree in the corresponding coefficient. For example choosing $\tau_2(1) = 2$ makes $[\sigma]_e$ and $[\psi_2(Q)]_e$ coincide in the second coefficient. Iterating this process greedily, we obtain the permutation $\tau_2 = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & * & 5 & 3 & 6 & 1 & * & * \end{smallmatrix}\right)$.

The undefined images are completed such that $\tau_2$ still induces an automorphism between $P$ and $\phi(P)$, e.g. $\tau_2 = \left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 4 & 5 & 3 & 6 & 1 & 8 & 7 \end{smallmatrix}\right)$.

Constructing $\psi_2(Q)$ with this greedy approach, it is both a signature that will verify with the same probability as a legitimately generated one and it is indistinguishable from $\sigma$ when evaluated on five of the eight possible Boolean tuples as

$$[\psi_2(Q)]_e = [1, -1, 0, 2, 2, -1, 0, 1].$$

When running the signature verification on $\sigma$ and the message corresponding to $Q$, $\sigma$ will evaluate to the value of a valid signature $5/8$ of the time. In the remaining $3/8$ of tuples, the sign of the evaluation may or may not be the same.

In the following, we generalise the example and we give the greedy algorithm which computes an automorphism $\psi$ in the solution set of Problem 4.1 such that $\psi(Q)$ is close to a given polynomial $\sigma$. The "distance" between the polynomials can be used to give a lower bound on the probability that $\sigma$ will be accepted as a signature. For the previous example, we say that the distance between $\psi_2(Q)$ and $\sigma$ is $3/8$. Notice that in general the smaller the distance, the bigger the probability $\sigma$ will be accepted during the verification.

### 5.1 Sufficiently good signatures

As discussed in Section 4.2, there are many different solutions $\psi$ to Problem 4.1 for each of which $\psi(Q)$ passes verification with the same probability as an honestly generated signature. To determine the probability with which a candidate signature $\sigma \in B_n$ passes verification, we would like to determine whether $\sigma$ is

sufficiently close to $\psi(Q)$ for *some* $\psi$ in the set of solutions to Problem 4.1, meaning $\psi(Q)$ and $\sigma$ evaluate to the same values on many Boolean tuples. We say $\psi(Q) \approx \sigma$ if the proportion of Boolean values where the evaluation of both polynomials coincides is large.

Note that given the signature verification this could be relaxed even further only demanding that $\psi(Q)$ and $\sigma$ evaluate to values with the same sign on many Boolean tuples.

Let $\phi$ be the sender's private key and recall that the polynomials $P$, as defined in Eq. (3), and $\phi(P)$ which can be computed from the public key evaluate to values in $\{0, 1, 2, \ldots, 63\}$ for the parameters proposed by the authors of BASS. Given $\sigma$, consider the following algorithm generalising Example 5.1. The algorithm computes an equivalent private key $\psi \in \mathrm{Aut}(B_n)$ in the set of solutions to Problem 4.1 such that $\psi(Q) \approx \sigma$.

1. Apply Theorem 4.2 to the polynomials $P$ and $\phi(P)$ as computed in Eq. (3) and compute the sets $S_i$, $T_i$ $\forall i \in \{0, 1, \ldots, 63\}$, i.e. the sets of Boolean tuples on which $P$, respectively $\phi(P)$, evaluate to $i$.
2. For every $i \in \{0, 1, \ldots, 63\}$, do:
   2.1. For each $c \in S_i$ (in any order), try to find $d \in T_i$ such that $Q(c) = \sigma(d)$. If one is found, take both of them out of their sets and define $\tau(c) = d$.
   2.2. For the remaining $c \in S_i$ for which we were not able to find a corresponding $d$, define $\tau(c) = d$ for any $d \in T_i$. Take both of them out of their sets until both sets are empty. This will happen as Theorem 4.2 guarantees $|S_i| = |T_i|$.
3. Output the automorphism $\psi$ corresponding to the computed permutation $\tau$, which we know to exist by *(v)* of Lemma 3.1.

As a consequence of Theorem 4.2, $\psi$ is a solution to Problem 4.1 because $\tau$ matches elements of $S_i$ with $T_i$. Using Eq. (1), we see that for any $\psi$ computed this way, we have $\psi(Q)(d) = Q(\tau^{-1}(d)) = Q(c)$. If $\tau(c)$ was defined during Step 2.1 of the algorithm, then $Q(c) = \sigma(d)$, i.e. $\psi(Q)(d) = Q(c) = \sigma(d)$.

Let $C$ denote the set of Boolean tuples $c$ such that $\tau(c)$ was defined on the Step 2.1 of the algorithm. Clearly $\psi(Q)$ and $\sigma$ are equal whenever evaluated on any Boolean tuple in $C$. Thus, computing a good $\psi$ and tracking the size of $C$ allows us to give a lower bound on the probability that a polynomial $\sigma \in B_{32}$ passes the signature verification of BASS.

*Estimating the probability a polynomial passes signature verification.* Let $u \in K_4$ be random and consider the corresponding polynomials

$$R := u(P_1, P_2, P_3, Q), \; S := u(\phi(P_1), \phi(P_2), \phi(P_3), \sigma) \in B_{32}.$$

Let $r$ be the proportion of positive values $R$ (and $S$) take when evaluated over the set $C$ where $\psi(Q)$ and $\sigma$ evaluate to the same values. Further, let $\delta := 1 - |C|/2^{32}$. Note that the set $C$ depends on the choices made in Step 2.1 when computing $\psi$ but its cardinality and thus $\delta$ is independent of these choices.

For the proportion of positive values $r_1$ of $R$ and $r_2$ of $S$ when evaluated over all Boolean tuples, we have

$$r_1 = (1-\delta)r + \delta p_1, \quad r_2 = (1-\delta)r + \delta p_2,$$

where $p_1$ and $p_2$ denote the proportions of positive values of $R$ and $S$, respectively, when evaluated over the complement of $C$. Clearly, we have $|r_1 - r_2| = \delta|p_1 - p_2|$. The number of positive values $N_1$ of $R$ and $N_2$ of $S$ when evaluated over 3000 Boolean tuples sampled during the signature verification follow a binomial distribution. Approximating both to a normal distribution as described in Section 3.1, their difference can be approximated by the normal distribution

$$N_1 - N_2 \sim N(3000(r_1 - r_2), 3000r_1(1-r_1) + 3000r_2(1-r_2)).$$

Clearly, the closer $r_1, r_2$ are to each other, the more likely the signature verification will succeed. In particular, suppose that $\sigma = \psi(Q)$ for some automorphism $\psi$ in the space of solutions of Problem 4.1. Then, it is easy to check that $\delta = 0$, which implies $r_1 = r_2$. Therefore, the probability of accepting $\sigma$ is the same as the one of a correctly generated signature.

### 5.2 Experimental observations

*Signing a message with its encoded digest.* We tried signing a message with its encoded digest $Q$ as a signature. The rational is that $Q$ and $\psi(Q)$ have the same images when evaluated over all Boolean tuples. Thus, one may hope that there are a number of Boolean tuples in $S_i$ and $T_i$ where $Q$ evaluates to the same value. The greater the number of such tuples, the larger the set $C$ and the smaller $\delta$. For the proposed parameters of BASS, we computed $\delta$ experimentally to be roughly $\approx 0.1$. Assuming $p_1, p_2$ are independent random variables following a uniform distribution, the expected value of $|r_1 - r_2|$ is then $\delta|p_1 - p_2| = \delta/3 \approx 0.033$. Using the result from the previous section, we can approximate

$$N_1 - N_2 \sim N(100, 1500).$$

Here, we used that the distribution has variance close to 1500 which follows from experiments suggesting that $r_i \approx 0.5$. Computing the values for the normal distribution, we get

$$\Pr(|N_1 - N_2| \leq 90) \approx 0.398.$$

This suggests that $Q$ itself should verify as a signature at least 40% of the time. Note that the assumptions made throughout were the worst case for an attacker.

Experimentally, using $Q$ as a signature turned out to be even more successful. For the proposed parameters, $Q$ was accepted between 78% and 94% of the time for most of the randomly chosen pairs of public keys and messages. A visualisation of our experiments is depicted in Fig. 1.

*Signing a message with* 0. Another idea is to sign a message with the zero-polynomial. In this case, the rational is that $Q$ evaluates to zero on many Boolean tuples and so will $\psi(Q)$. Choosing random messages, we computed $\delta \approx 0.87$ with respect to $\sigma = 0 \in B_n$. Our experiments then showed that the probability of accepting 0 as a valid signature tends to be larger than 0.1 for the parameters proposed for BASS. This highlights a clear weakness in the signature validation and yields a trivial attack on BASS.

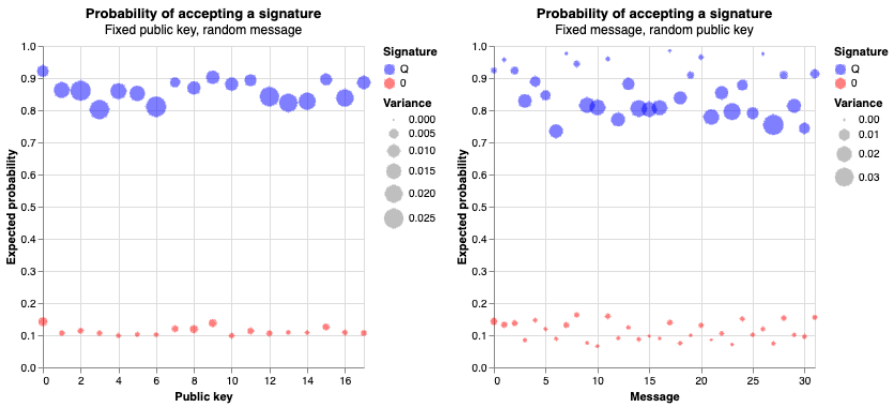


**Fig. 1.** Given randomly chosen pairs of public key and message, we estimated the probability of accepting $Q$ or 0 as a valid signature for each pair. The probabilities were approximated by running the verification algorithm 100 times for all $(pk, \mathfrak{m})$-pairs, counting how often the verification succeeded. Mean and variance of the probabilities are computed for all the pairs $(pk, \mathfrak{m})$ that share either the public key (left) or the message (right).

Both attacks, signing with $Q$ as well as signing with 0, exploit the low accuracy of BASS's probabilistic verification algorithm. Clearly, the success probability of the attacks could be lowered when checking that the proportions of positive values in $R$ and $S$ diverge by less than 3%. When reducing this error bound in the signature verification, either even more correctly generated signatures would be rejected or one would need to evaluate $R$ and $S$ on a much larger number of Boolean tuples.

Assume we accepted the current probability of rejecting a legitimately generated signature 2% of the time. Further, assume that the attack of signing a message with its encoded hash $Q$ succeeds only with a probability of 78% after evaluating 3000 Monte Carlo samples. Note this was among the lowest success rates observed in the experiments and the attack performed on average much better. Even under these very optimistic assumptions (for a signer), it would be necessary to sample $\approx 295.000$ Boolean tuples during signature verification in order for the attack of signing with the encoded hash of a message $Q$ to succeed with probability less than $2^{-128}$. Since signature verification is linear in the

number of Boolean tuples checked, this would slow down signature verification almost by 100x.

*Remark 5.2.* Given the inconsistent description of the signature verification of BASS, as mentioned in Remark 3.1, we also ran the trivial attacks of signing a message directly with its hashed encoding $Q$ or with 0 for the signature verification which compares the number of zeroes instead of the number of positive values. The success probability of the attacks was even higher with $Q$ being accepted as a valid signature for almost all message and signature pairs that we tested. The results are depicted in Appendix B.

## 6 Recovering and signing with equivalent private keys

In this section, we provide a key-only attack which computes an equivalent private key, i.e. a solution to Problem 4.1, given only the public key of the signature scheme. We first provide an attack using a first-in, first-out (FIFO) criterion. Then, we speculate about a hypothetical countermeasure and we suggest a more sophisticated attack that will succeed in this case. Asymptotically our attack has the same complexity as the signing algorithm of BASS.

Throughout this section, we let $\psi$ be an equivalent private key that we aim to compute and we represent automorphisms by their action on the orthogonal basis.

### 6.1 FIFO attack

**Computing an equivalent key.** As observed in Section 4.2, we can find a solution $\psi$ to Problem 4.1 by mapping elements in $S_i$ to elements in $T_i$ where $S_i$ and $T_i$ are defined with respect to the polynomials $P$ and $\phi(P)$ from Eq. (3). We know that $P$ takes values in $\{0, 1, \ldots, 63\}$ for the proposed parameters in BASS. Given two polynomials $A, B \in B_n$ taking values in $\{0, 1, \ldots, 63\}$ such that there exists an automorphism that maps one polynomial to the other, Algorithm 1 describes how to recover an automorphism with the same action on the polynomials using a first-in, first-out approach.

**Lemma 6.1** *Algorithm 1 is correct and runs with a time and memory complexity of* $O(\text{poly}(n)2^n)$.

*Proof.* The correctness of the algorithm follows from Lemma 3.1($v$) and Theorem 4.2. Computing $S_i$ and $T_i$ from polynomials given in the orthogonal basis takes time $O(\text{poly}(n)2^n)$ by once passing through the polynomial's coefficients. Transforming polynomials from the standard basis to the orthogonal basis can be done with a divide-and-conquer recurrence running in $O(m \log(m))$, where $m$ is the size of the given polynomial in the standard basis which is bounded by $2^n$. Thus, the translation from standard to orthogonal basis can be achieved in time $O(\text{poly}(n)2^n)$. The second loop touches upon all Boolean tuples $c \in \{0, 1\}^n$ exactly once and thus runs in time $O(\text{poly}(n)2^n)$.

**Algorithm 1** FIFO

**Require:** $[A]_e, [B]_e \in B_n$ such that there exists $\phi \in \mathrm{Aut}(B_n)$ with $A = \phi(B)$
**Ensure:** $\psi \in \mathrm{Aut}(B_n)$ with $\psi(A) = B$
  **for** $i \in \{0, \ldots, 63\}$ **do**
    Compute $S_i = \{c \in \{0,1\}^n \mid A(c) = i\}$
    Compute $T_i = \{c \in \{0,1\}^n \mid B(c) = i\}$
  **end for**
  **for** $i \in \{0, \ldots, 63\}$ **do**
    **for** $j \leftarrow 1$ to $|S_i|$ **do**
      $c \leftarrow S_i[j]$
      $d \leftarrow T_i[j]$
      $\psi(e_c) \leftarrow e_d$
    **end for**
  **end for**
  **return** $\psi$

The union of the sets $S_i$ and $T_i$ are all Boolean $n$-tuples whose storage requires $O(n2^n)$ bits of memory. Similarly, the automorphism can be stored in a representation that describes its action on each element of the orthogonal basis requiring $O(n2^n)$ bits of memory. □

**Evaluating the equivalent key $\psi$ on $Q$.** Having computed an equivalent key $\psi$ in the set of solutions to Problem 4.1 by its action on the orthogonal basis in Algorithm 1, we are left with computing $[\psi(Q)]_b$ to sign a message.

We do so by expressing $[Q]_e$ in the orthogonal basis, computing $[\psi(Q)]_e$ and then converting it to the standard basis as summarised in Algorithm 2. We claim that this can be done in $O(\mathrm{poly}(n)2^n)$ time, using $O(\mathrm{poly}(n)2^n)$ memory.

**Algorithm 2** Signing

**Require:** $[\psi]_e$ an equivalent private key, $\mathfrak{m}$ a message
**Ensure:** A signature $[\psi(Q)]_b$ of $\mathfrak{m}$
  $[Q]_b \leftarrow \mathrm{Encode}(H(\mathfrak{m}))$ ▷ Standard hash and encoding
  $[Q]_e \leftarrow \mathrm{StandardToOrthogonal}([Q]_b)$
  $[\psi(Q)]_e \leftarrow \psi([Q]_e)$
  $[\psi(Q)]_b \leftarrow \mathrm{OrthogonalToStandard}([\psi(Q)]_e)$
  **return** $[\psi(Q)]_b$

To run Algorithm 2, we need an efficient algorithm to transform a polynomial's representation with respect to the standard basis to the orthogonal basis and vice versa. As claimed in the proof of Lemma 6.1, the first transformation can be done with a divide-and-conquer recurrence in time $O(\mathrm{poly}(n)2^n)$. To transform a polynomial $A$ from the orthogonal basis to the standard basis we apply the following relation.

$$A = \sum_{c\in\{0,1\}^n} A(c)e_c = (1-x_n)\sum_{c\in\{0,1\}^{n-1}} A(c\|0)e_c + x_n \sum_{c\in\{0,1\}^{n-1}} A(c\|1)e_c$$
$$= A_0 + x_n(A_1 - A_0),$$

where

$$A_i = \sum_{c\in\{0,1\}^{n-1}} A|_{x_n=i}(c)e_c.$$

Constructing $A_0$ and $A_1$ requires simply selecting the terms $A(c)e_c$ in $A$ with $c_n$ equal to 0 or 1, respectively. Computing and subtracting the polynomials thus takes time linear in the length of $A$. Since the polynomials $A_i$ do not contain any terms with $x_n$, the multiplication only appends $x_n$. Conducting the computation recursively, the standard basis representation can therefore be computed from the orthogonal basis in time $O(\text{poly}(n)2^n)$ as well.

As before, we store the automorphism $\psi$ as its action on the orthogonal basis which takes $O(n2^n)$ bits. Storing $[Q]_b$ requires $O(n)$ memory as $Q$ consists of at most $3n$ monomial terms. Computing the representation of $Q$ in the orthogonal basis requires $O(\text{poly}(n)2^n)$ bits of memory. Evaluating $\psi$ on $[Q]_e$ is just a reordering of the basis elements and the recursion to compute $Q$ in the standard basis requires the same amount of memory in every step. As such the memory cost of signing messages with the equivalent private key given with respect to its action on the orthogonal basis is $O(\text{poly}(n)2^n)$.

Note that when a signature is honestly computed, a signer has to compute polynomial products of the form $\phi(x_i)\cdot\phi(x_j)$ for some indices $i \neq j$. In general, this has time complexity $O(\text{poly}(n)2^n)$ as well.

We implemented and verified the correctness of the attack in Python and ran it on several instances for $n = 28$, small enough to run the attack with the 22GB of memory available on our laptop [2].

### 6.2 Further improvements of the attack

We have seen how to compute an equivalent secret key from the public key provided using Algorithm 1, and how this can be used to forge signatures that will pass verification with the same probability as a legitimately generated signature. This raises the question:

*Are our forgeries indeed indistinguishable from legitimate signatures?*

It turns out that forgeries computed with an equivalent key given by Algorithm 1 will usually be polynomials with significantly more terms than honestly generated ones, when expressed in the standard basis. This is an artifact of the way private keys are generated in BASS. Instead of choosing private keys uniformly at random in $\text{Aut}(B_{31})$ and then extending them to an automorphism in $\text{Aut}(B_{32})$, the secret automorphism $\phi$ is sampled in a way such that $\phi(x_i) = x_j$ for some $j$ for approximately one-quarter of the indices $i \in \{1,\ldots,31\}$. This is a

property of the secret keys we have not yet exploited and that is not present in our equivalent keys in general. To address a hypothetical countermeasure which would reject signatures of length surpassing a certain threshold, we discuss how we can find equivalent keys that generate more compact signatures.

In this subsection, we briefly describe how to predict that $\psi(x_i) = x_j$ for some $i, j \in \{1, \dots, 32\}$ and how to adapt Algorithm 1 to incorporate these predictions. Clearly, the more pairs of indices $i, j$ with $\psi(x_i) = x_j$ we find for a solution $\psi$ to Problem 4.1, the shorter we can expect the resulting signatures to be in the standard basis.

We start with the following observation.

**Lemma 6.2** *Let $P$, $\phi(P)$ as defined in Eq. (3), and $S_i$ and $T_i$ bet the sets of Boolean tuples where $P$ and $\phi(P)$, respectively, evaluate to the same value $i$ as defined in Theorem 4.2. Further, define*

$$S_{i,k,0} := S_i \cap \{c \in \{0,1\}^n \mid c_k = 0\}, \quad T_{i,\ell,0} := T_i \cap \{c \in \{0,1\}^n \mid c_\ell = 0\}.$$

*If there exist $k, \ell \in [n]$ such that $|S_{i,k,0}| = |T_{i,\ell,0}|$ for all $i \in \{0, 1, \dots, m-1\}$, then there exist $\psi_0, \psi_1 \in \mathrm{Aut}(B_{n-1})$ such that*

$$\psi_0(P|_{x_k=0}) = \phi(P)|_{x_\ell=0}, \quad \psi_1(P|_{x_k=1}) = \phi(P)|_{x_\ell=1}.$$

*For $\psi \in \mathrm{Aut}(B_n)$ defined as*

$$\psi(x_k) = x_\ell, \ \ \psi(x_i) = (1 - x_\ell)\psi_0(x_i) + x_\ell\psi_1(x_i), \ \ \forall i \neq k$$

*we then have $\psi(P) = \phi(P)$.*

*Conversely, if $\psi \in \mathrm{Aut}(B_n)$ such that $\psi(P) = \phi(P)$ and $\psi(x_k) = x_\ell$ for some $k, \ell \in [n]$, then $|S_{i,k,0}| = |T_{i,\ell,0}|$ for all $i \in \{0, 1, \dots, m-1\}$.*

The proof of Lemma 6.2 is located in Appendix A. The lemma allows us to split the problem of finding $\psi \in \mathrm{Aut}(B_n)$ into to two smaller problems where we have to find $\psi_0, \psi_1 \in \mathrm{Aut}(B_{n-1})$. By iterating this process, we may find several indices in the standard basis where $\psi$ has a very simple image, i.e. $i, j$ such that $\psi(x_i) = x_j$. For example, suppose that when computing $\psi_0, \psi_1$, we have $\psi_0(x_i) = x_j = \psi_1(x_i)$. Then, by definition of $\psi$ we have

$$\begin{aligned}\psi(x_k) &= x_\ell \\ \psi(x_i) &= (1 - x_\ell)x_j + x_\ell x_j = x_j.\end{aligned}$$

That is $\psi$ has two indices with simple images with respect to the standard basis. Note that the secret automorphism of a signer $\phi$ has many simple images, thus by the converse direction we know that we can find $\psi_0, \psi_1$ that will satisfy the necessary condition of the lemma. Yet, we may find false positives in the sense that the necessary condition of Lemma 6.2 can be fulfilled even if $\phi(x_k) \neq x_\ell$ for the secret automorphism $\phi$.

*Remark 6.1.* Any automorphism which sends $x_k$ to $x_\ell$ can be obtained using the reverse direction of Lemma 6.2. In other words, all automorphisms $\psi \in \mathrm{Aut}(B_n)$ with $\psi(x_k) = x_\ell$ can be obtained by merging solutions for $\psi_0, \psi_1 \in \mathrm{Aut}(B_{n-1})$.

We implemented a variant of our attack which recursively finds a list of valid predictions by computing and matching the sets $S_{i,k,0}$ and $T_{i,k,0}$ from Lemma 6.2. When no further predictions for simple images can be made, we call Algorithm 1 previously described.

Making $r$ valid predictions breaks the problem of finding an equivalent private key $\psi$ into $2^r$ smaller problems of size $n-r$. Each problem can be solved in time $O(\text{poly}(n)2^{n-r})$. The necessary predictions can be made in time $O(\text{poly}(n)2^n)$. Overall the time and memory complexity thus remains at $O(\text{poly}(n)2^n)$. However, the automorphism found will have more simple images with respect to the standard basis, making the resulting forgeries more compact in the standard basis.

## 7 Conclusion

In this paper we describe multiple shortcomings of and attacks on the BASS signature scheme by Grigoriev, Ilmer, Ovchinnikov and Shpilrain [1].

We explain two ways of generating trivial forgeries which pass the verification with high probability for the proposed parameters. Generating these forgeries takes less time than legitimately generating a signature. The attack could be prevented by changing the parameters of the probabilistic signature verification. However, even under very optimistic assumptions this would lead to a roughly 100x slowdown in signature verification.

Further, we give a key-only attack which recovers an equivalent private key from the public key. The complexity of this attack is asymptotically the same as honestly running the signature scheme.

Along the way, we contradict several claims by the authors of BASS. We show that private keys are far from being (almost) unique, and we show that the scheme's signature verification leads to significantly more false negatives for the proposed parameters than claimed.

Due to the numerous design flaws and significant security issues of BASS identified in this paper, the signature scheme and in particular its probabilistic signature verification appear not to be fit for use.

## Bibliography


[1] Dima Grigoriev, Ilia Ilmer, Alexey Ovchinnikov, and Vladimir Shpilrain. Bass: Boolean automorphisms signature scheme. In Mark Manulis, Diana Maimuţ, and George Teşeleanu, editors, *Innovative Security Solutions for Information Technology and Communications*, pages 1–12, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-52947-4.

[2] A. Rodríguez. Python implementation of attacks on BASS, 2024. URL https://github.com/AlexRGO3/BASS_attack.

## A Proofs

*Proof (Lemma 3.1).* For *(i)* notice that any polynomial expressed in the standard basis can be written as a polynomial in the orthogonal basis. Given $c \in \{0,1\}^n$, define $I_1 := \{i : c_i = 1\}$, $I_0 := \{i : c_i = 0\}$. Then

$$b_c = \prod_{i \in I_1} x_i = \prod_{i \in I_1} x_i \prod_{j \in I_0} [x_j + (1 - x_j)] = \sum_{c\,:\,c_i = 1\ \forall i \in I_1} e_c.$$

Uniqueness will follow with the proof of *(iii)*.

*(ii)* As $c \neq d$, there exists an $i \in [n]$ such that $c_i \neq d_i$. If $c_i = 0$, then

$$e_c = (1 - x_i) \cdot \prod_{j \neq i} x_j^{c_j} (1 - x_j)^{1 - c_j}.$$

If $c_i = 1$, then

$$e_c = x_i \cdot \prod_{j \neq i} x_j^{c_j} (1 - x_j)^{1 - c_j}.$$

In both cases, evaluation at $d$ is zero. Using the same notation, we have

$$e_c \cdot e_d = (1 - x_i) \cdot x_i \prod_{j \neq i} x_j^{c_j} (1 - x_j)^{1 - c_j} \cdot \prod_{j \neq i} x_j^{d_j} (1 - x_j)^{1 - d_j} = 0,$$

where we used $(1 - x_i)x_i = x_i - x_i^2 = 0$. Finally, we have

$$e_c(c) = \prod_{i=1}^{n} 1 = 1$$
$$e_c \cdot e_c = \prod_{i=1}^{n} x_i^{2c_i} (1 - x_i)^{2(1 - c_i)} = \prod_{i=1}^{n} x_i^{c_i} (1 - x_i)^{1 - c_i} = e_c$$

using again that $x_i^2 = x_i$.

*(iii)* From *(i)* we know that any polynomial $A \in B_n$ can be written as

$$A = \sum_{c \in \{0,1\}} A_c e_c$$

for some integers $A_c$. Evaluating both sides at $d \in \{0,1\}^n$ and using *(ii)*, we obtain the equality $A(d) = A_d$. Hence,

$$A = \sum_{c \in \{0,1\}} A(c) e_c$$

which is a unique expression.

*(iv)* Let $\phi \in \mathrm{Aut}(B_n)$. By $\phi$'s linearity and the previous results, we have

$$\phi(e_d) = \sum_{c \in \{0,1\}^n} \phi(e_d)(c) e_c$$
$$\phi(e_d) \cdot \phi(e_d) = \sum_{c \in \{0,1\}^n} \phi(e_d)(c)^2 e_c$$
$$\phi(e_d) = \phi(e_d \cdot e_d) = \phi(e_d) \cdot \phi(e_d)$$

In particular, we have $\phi(e_d)(c) = \phi(e_d)(c)^2$, i.e. 0 or 1. Thus, we can write

$$\phi(e_d) = \sum_{c \in S_d} e_c$$

for some non-empty subset $M_d$. Note $M_d$ is non-empty as $\phi$ must have an inverse. For $d' \in \{0,1\}^n$, $d' \neq d$, we have

$$\phi(e_d) \cdot \phi(e_{d'}) = \sum_{c \in M_d \cap M_{d'}} e_c$$
$$\phi(e_d) \cdot \phi(e_{d'}) = \phi(e_d \cdot e_{d'}) = \phi(0) = 0,$$

i.e. $M_d \cap M_{d'} = \emptyset$. Here, we used that the expression of a polynomial in base $e_c$ is unique and the coefficients of the 0 polynomial are all zero. The $2^n$ sets $M_d$ are contained in the power set of $\{0,1\}^n$ are all non-empty and disjoint and thus all must have size 1. Define $\pi$ such that $M_d = \{\pi(d)\}$. We observe that $\pi$ is a permutation. Further, $\phi(e_d) = e_{\pi(d)}$, which we wanted to prove.

*(v)* Let $\pi$ be a permutation and define a morphism $\phi$ with

$$\phi(x_i) = \sum_{\substack{c \in \{0,1\}^n \\ c_i = 1}} e_{\pi(c)}.$$

We can verify that

$$\phi(x_i)(\pi(d)) = \sum_{\substack{c \in \{0,1\}^n \\ c_i = 1}} e_{\pi(c)}(\pi(d)).$$

As $\pi$ is a permutation and using *(ii)*, we conclude that $e_{\pi(c)}(\pi(d)) = e_c(d)$. Therefore,

$$\phi(x_i)(\pi(d)) = \begin{cases} 1 & \text{if } d_i = 1, \\ 0 & \text{if } d_i = 0. \end{cases} = d_i.$$

Then,

$$\phi(e_c) = \prod_{i=1}^{n} \phi(x_i)^{c_i} (1 - \phi(x_i))^{1-c_i}.$$

Evaluating this expression at $\pi(d)$, we have

$$\phi(e_c)(\pi(d)) = \begin{cases} 1 & \text{if } d = c, \\ 0 & \text{otherwise} \end{cases}.$$

Using *(iii)*, we get $\phi(e_c) = e_{\pi(c)}$. Thus, $\phi$ is a morphism with the desired property. Defining $\phi^{-1}$ analogously using the permutation $\pi^{-1}$, we see that $\phi$ is indeed an automorphism as it has an inverse. □

*Proof (Lemma 6.2).* The existence of $\psi_0$ and $\psi_1$ follows by applying Theorem 4.2 for $A := P|_{x_k=0}$ and $B := \phi(P)|_{x_\ell=0}$. The sets of Boolean tuples where $A$ and $B$ evaluate to $i$ are $S_{i,k,0}$ and $T_{i,\ell,0}$. Similarly, for $A' = P|_{x_k=1}$ and $B' = \phi(P)|_{x_\ell=1}$, the corresponding sets are $S_i \setminus S_{i,k,0}$ and $T_i \setminus T_{i,\ell,0}$.

Note that $\psi$ as defined in the lemma is a morphism as it extends linearly, and it is an automorphism because $\psi(e_c) = e_{\pi(c)}$ for some permutation $\pi$. Next, we show that $\psi(P) = \phi(P)$. Consider any $c \in \{0,1\}^n$. If $c_\ell = 0$, we have

$$\begin{aligned}
\psi(P)(c) &= P(\psi(x_1), \ldots, \psi(x_n))(c) \\
&= P((1-x_\ell)\psi_0(x_1) + x_\ell\psi_1(x_1), \ldots, x_\ell, \ldots, (1-x_\ell)\psi_0(x_n) + x_\ell\psi_1(x_n))(c) \\
&= P(\psi_0(x_1), \ldots, 0, \ldots, \psi_0(x_n))(c) = P(\psi_0(x_1), \ldots, \psi_0(x_n))|_{x_k=0}(c) \\
&= \psi_0(P|_{x_k=0})(c) = \phi(P)|_{c_\ell=0}(c) = \phi(P)(c).
\end{aligned}$$

Similarly, if $c_\ell = 1$

$$\begin{aligned}
\psi(P)(c) &= P(\psi(x_1), \ldots, \psi(x_n))(c) \\
&= P((1-x_\ell)\psi_0(x_1) + x_\ell\psi_1(x_1), \ldots, x_\ell, \ldots, (1-x_\ell)\psi_0(x_n) + x_\ell\psi_1(x_n))(c) \\
&= P(\psi_1(x_1), \ldots, 1, \ldots, \psi_1(x_n))(c) = P(\psi_1(x_1), \ldots, \psi_1(x_n))|_{x_k=1}(c) \\
&= \psi_1(P|_{x_k=1})(c) = \phi(P)|_{c_\ell=1}(c) = \phi(P)(c).
\end{aligned}$$

By *(iii)* of Lemma 3.1 and as $\psi(P)(c) = \phi(P)(c)$ for all $c \in \{0,1\}^n$, it follows that $\psi(P) = \phi(P)$.

For the other direction of the proof, suppose there exists $\psi \in \mathrm{Aut}(B_n)$ such that $\psi(P) = \phi(P)$ and $\psi(x_k) = x_\ell$ for some $k, \ell \in [n]$. Grouping terms in the orthogonal basis, we obtain

$$P = x_k p_1 + (1 - x_k) p_0$$

where $p_0$ and $p_1$ do not depend on $x_k$. Evaluating on both sides we get $p_0 = P|_{x_k=0}$ and $p_1 = P|_{x_k=1}$. Applying $\psi$ to both sides, we get

$$\phi(P) = \psi(P) = x_\ell \psi(p_1) + (1 - x_\ell)\psi(p_0).$$

Evaluating on $x_\ell = 0$, we get

$$\phi(P)|_{x_\ell=0} = \psi(p_0)|_{x_\ell=0}.$$

We can define $\psi_0$ by linear extension as $\psi_0(x_i) = \psi(x_i)|_{x_\ell=0}$ for all $i \neq k$, i.e. we have $\phi(P)|_{x_\ell=0} = \psi_0(p_0)$. In particular, $\psi_0$ is an automorphism that maps $P|_{x_k=0}$ to $\phi(P)|_{x_\ell=0}$ and thus $|S_{i,k,0}| = |T_{i,\ell,0}|$ for all $i \in \{0, 1, \ldots, m-1\}$ by Theorem 4.2. □

## B Success probability of trivial attacks with alternative verification

The following plots depict the success probability of our trivial attacks where one signs with the encoded hash of the message $Q$ or the zero-polynomial, when the number of zeroes (instead of the number of positive values) are counted during signature verification. The public key and message pairs used are the same as used in the experiments to generate Fig. 1.

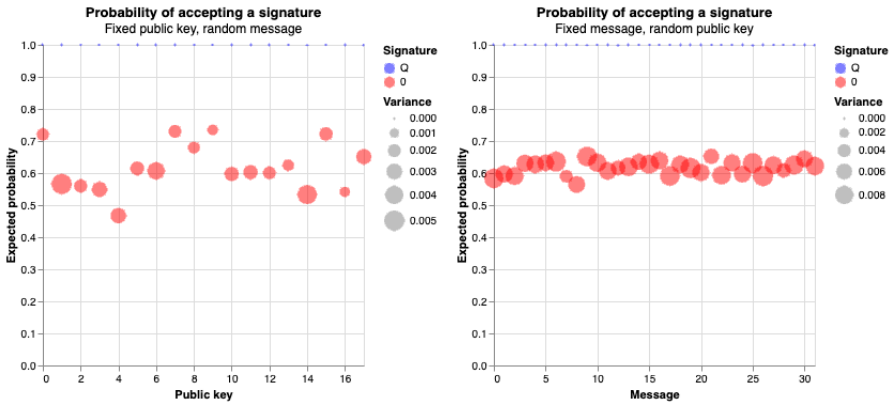


**Fig. 2.** Given randomly chosen pairs of public key and message, we estimated the probability of accepting $Q$ or 0 as a valid signature for each pair. The probabilities were approximated by running the verification algorithm 100 times for all $(pk, \mathfrak{m})$-pairs, counting how often the verification succeeded when the number of zeroes was compared. Mean and variance of the probabilities are computed for all the pairs $(pk, \mathfrak{m})$ that share either the public key (left) or the message (right).