Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto

Christoph Dobraunig[1], Daniel Kales[2], Christian Rechberger[2], Markus Schofnegger[2], and Greg Zaverucha[3]

[1]Lamarr Security Research, Graz, Austria
[2]IAIK, Graz University of Technology, Graz, Austria
[3]Microsoft Research, Redmond, WA, USA

May 26, 2021

Abstract

So far, signature schemes based on the MPC-in-the-head approach (MPCitH) have either been designed by taking a proof system and selecting a suitable symmetric-key primitive (Picnic, CCS16), or starting with an existing primitive such as AES and trying to find the most suitable proof system (BBQ, SAC19 or Banquet, PKC21). In this work we do both: we improve certain symmetric-key primitives to better fit signature schemes, and we also propose a new signature scheme by *co-designing* a proof system and a new block cipher. Our concrete results are as follows.

First, we show how to provably remove the need to include the key schedule of block ciphers. This simplifies schemes like Picnic and it also leads to the fastest and smallest AES-based signatures. For example, we achieve signature sizes of around 10.8 to 14.2 KB for the 128-bit security level, on average 10% shorter than Banquet and 15% faster.

Second, we investigate a variant of AES with larger S-boxes we call LSAES, for which we can argue that it is very likely at least as strong as AES, further reducing the size of AES-based signatures to 9.9 KB.

Finally, we present a new signature scheme, Rainier, based on a new block cipher called Rain combined with a Banquet-like proof system. To the best of our knowledge, it is the first MPCitH-based signature scheme which can produce signatures that are less than 5 KB in size; it also outperforms previous Picnic and Banquet instances in all performance metrics.

1 Introduction

Digital signatures are one of the fundamental primitives in cryptography and also play a critical role in today's internet infrastructure. Most currently used signature schemes rely on the discrete logarithm assumption in elliptic curve groups, or the RSA assumption. While these hardness assumptions are believed to hold against classical attackers, the situation might change with the introduction of powerful quantum computers.

As of today, no quantum computer threatening currently deployed cryptography exists. Still, the cryptographic community is already searching for public-key primitives which are secure against quantum attacks. Candidates for such primitives in the ongoing NIST post-quantum standardization project include schemes based on the hardness of lattice problems (e.g., [LDK+20, PFH+20]), problems from coding theory [ABC+20], the hardness of solving multivariate quadratic equation systems [DCP+20] or different assumptions related to finding isogenies between elliptic curves [JAC+20].

In addition to the above hardness assumptions, digital signatures can also be built solely using symmetric primitives, which has the advantage of relying on well-studied primitives instead of more "structured" hardness assumptions. Here, two examples are SPHINCS+ [BHK+19, HBD+20], relying only on hash functions, and Picnic [CDG+17, ZCD+20], relying on hash functions and a block cipher.

The constructions in the Picnic family follow an interesting design approach: The secret key of the signature scheme is a block cipher secret key k, with the public key being a (plaintext, ciphertext) pair (p, c). A signature is a non-interactive zero-knowledge proof of knowledge (ZKPoK) of the secret key for the public (plaintext, ciphertext) pair, with the message being included in the challenge generation of the proof. Importantly, the ZKPoK also has to provide security against quantum attackers; a class of zero-knowledge proofs that fulfill this requirement are the ones based on the MPC-in-the-head (MPCitH) paradigm [IKOS07]. In these proofs, the prover simulates a multiparty computation protocol executing the function to be proven for a number of parties, commits to the internal state of all parties during the protocol, and is then later challenged to open a subset of the parties' states to the verifier. If the internal state of the opened parties is consistent with a real execution, then the verifier gains some assurance that the execution was valid, with multiple parallel repetitions being executed to boost the soundness.

In Picnic, the block cipher is LowMC [ARS+15], which has the design goal of being efficiently computable in MPC protocols. This means that the internal state that needs to be committed to and revealed can be smaller and therefore, the overall size of the signature is reduced. This is accomplished by having a low multiplicative complexity, as multiplications (AND gates) are the most expensive part of MPC protocols that use linear secret sharing. This is a delicate balance, as the nonlinear operations of a block cipher are essential for security. LowMC is also a relatively new design, and (like all ciphers) is not as well analyzed as AES [DR20]. While AES would be an obvious first choice as a block cipher to be used in a Picnic-style signature, its Boolean circuit representation (which is used in the ZKPoK protocols in Picnic, ZKB++/ZKBoo [CDG+17, GMO16], and KKW [KKW18]) is much larger than that of LowMC, leading to signatures of about 209 KB for ZKB++ [GCZ16], and 52 KB for KKW [dDOS19].

Recently, a line of work has started to improve these AES-based signatures. In BBQ [dDOS19], the authors modify the MPC protocol to execute a circuit over the AES-native field $\mathbb{F}_{2^8}$ (rather than a binary circuit) and show how to improve the computation of the inverse in the AES S-box. This results in signatures of about 30 KB in size, which is a good improvement, but still large compared to the 12 KB signatures that Picnic using LowMC can achieve. In a recent follow-up work called Banquet [BdSGK+21], the authors proposed a new MPCitH protocol that reduces signature sizes even further by allowing the prover to inject known values into the MPC computation, removing the need to perform the computation of the inverse in the MPC protocol. The verifier needs to check the validity of the injected values, but this is done using a polynomial checking protocol which can prove the validity of the hundreds of inversions in AES efficiently in a batched manner. The final signature size for Banquet signatures ranges from 13.2 to 20 KB, which is close to the sizes of Picnic. However, the instances with smaller signatures have significantly slower signing and verification times than Picnic.

When it comes to designing ciphers for MPC-in-the-head proofs, Banquet has shown that field inversion can be more efficient than expected, especially when measuring the amount of non-linearity per unit of proof size. Since field inversion is a choice non-linear operation in block cipher design (e.g, it is the *only* non-linear operation in AES), a natural question is whether we can design a cipher to make use of this efficiency directly. We further observe that performance of the Banquet proof system is naturally much better when the inverse operations are in larger fields; which points to block cipher designs with large S-boxes. Inversion over large fields (we consider $\mathrm{GF}(2^{32})$ to $\mathrm{GF}(2^{256})$) would be a non-starter for traditional block ciphers, were performance is expected to be a handful of cycles per byte. However, in our MPCitH setting these inversions are comparatively cheap to prove. In summary, cipher designs using inversions in large fields have the potential to provide abundant non-linear operations, with record-low MPCitH costs.

Contributions. In this work, we investigate three methods of reducing MPCitH signature sizes further, while simultaneously improving the performance of signing and verification. Our results cover a range of options from more conservative (but less performant), to more performant (but with stronger assumptions).

- We investigate the use of AES as a public permutation in a single-key Even–Mansour construction. The use of a public constant for the AES key removes the need to calculate the AES key schedule as part of the MPC protocol, reducing the number of S-boxes (and therefore inversions) from 200 to 160 for AES-128, which leads to smaller signatures using the Banquet protocol.
- In the Banquet protocol, the prover-provided injected values are lifted from the AES field $\mathbb{F}_{2^8}$ to a larger field $\mathbb{F}_{2^{8\lambda}}$ to reduce the soundness error of the validity check. This step leads to an increase in signature size, since elements of that larger field are included in the final signature. We investigate the security of a variant of AES that uses 32-bit S-boxes and show that we can then remove the lifting step, leading to smaller signatures.
- Finally, we present a new block cipher, called Rain (Random Affine Inverse Nyberg-inspired), that is ideally suited for the MPCitH protocol. Rain follows a very simple design; having no key schedule and each round only consisting of a constant addition, a matrix multiplication and a field inversion. We then analyze its security as a one-way function, and finally build a signature algorithm called Rainier, using Rain and a simplified variant of the Banquet proof system. Rainier is, to the best of our knowledge, the first MPCitH-based signature scheme with signatures less than 5 KB

in size and it also outperforms previous Picnic and Banquet instances in all performance metrics.

A side benefit of the EM and Rain constructions is that signature private keys may be sampled uniformly whereas in BBQ and Banquet a negligible part of the key space must be excluded.

We have implemented all of these signature scheme variants, and provide detailed comparisons of running time, signature size and also briefly compare to other post-quantum signatures (not based on the MPCitH paradigm). Our implementation is available at `https://github.com/IAIK/rainier-signatures`.

Cryptanalysis Scenario. In our scenario, an attacker is only able to see a single (plaintext, ciphertext) pair, and security of the signature scheme requires only that the block cipher is a one-way function of the key. This is given from the fact that the public key of the signature scheme is a single, fixed, random plaintext-ciphertext pair and an attacker does not get access to any other pairs under the same key from the signer. Security claims in this paper should be intpreted in this scenario. This naturally prevents a large class of statistical attacks, since they usually require multiple pairs. For example, at least two pairs are necessary for a classical differential attack.

Related Work to Rain. Our proposal Rain described in Section 4 shows several similarities to MiMC [AGR+16] and the Marvellous designs Jarvis [AD18] and (instances of) *Vision* [AAB+20]. These designs also aim to minimize a certain cost metric in arithmetization-oriented scenarios, and all three of them are built using a single large S-box covering the full permutation state.

The round function of Rain is composed of two very different building blocks. First, we use the inversion in the field F_{2^n}, arguably a very structured operation. The second part is then an unstructured random affine layer with operations in F_2. The latter building block together with the fact that our proof system can efficiently handle these two rather different types of operations is crucial. This is nicely illustrated when looking at necessary round numbers for example at the 128-bit security level. For MiMC and its primary use cases, operations are preferred to all be in the same field and hence such an affine layer was not used at all, leading to a requirement of more than 80 rounds for security. For Jarvis/*Vision* this is reduced to 36 and without safety margin could be halved to 18 rounds. The main reason for this reduction is that a *structured* affine layer is introduced in every round and shown to allow for efficient implementations. With the *unstructured* affine layer in Rain, our analysis suggests that 3 rounds are already sufficient, leading to substantial efficiency improvements despite the lack of structure. This is also supported by Nyberg's analysis [Nyb94] showing that finite field inversion has a large distance from affine functions.

Related Work to Large S-Boxes. Using S-boxes or non-linear permutations working on more than eight bits is not new. In the early 1990s, we see for example the Knudsen–Nyberg cipher [NK93] using cubing in GF(2^{37}), or Subterranean [CDGP93] using the χ-layer [Dae95] on 257 bits. Also in recent cipher designs, we see the use of fairly large S-boxes. For example, Rasta [DEG+18] and Subterranean 2.0 [DMMR20] use large versions of the

χ-layer, while Alzette [BBdS+20] is an S-box design for 64-bit inputs/outputs based on modular additions, rotations and XORs. Other examples include MiMC [AGR+16], GMiMC [AGP+19], HadesMiMC [GLR+20], Poseidon [GKR+21], and Marvellous [AAB+20] instances with large S-boxes having a very simple algebraic description.

Related Work to MPCitH Proof Systems for AES. Concurrently, de Saint Guilhem et al. proposed Limbo [dSGOT21], a proof system for generic circuits. Buiding on the ideas of Banquet [BdSGK+21], the prover injects the output value of all multiplication gates into the circuit and the validity of the injected values is checked in the MPC protocol. While suitable for generic circuits, their approach can also be used to build signature schemes with runtimes and sizes very close to Banquet. A natural implication of our work on single-key Even-Mansour variants of AES is to investigate the performance of EM-AES signatures in Limbo. Based on the formulas given in [dSGOT21], we expect similar performance to our Banquet variants, with Limbo having slightly larger signatures and slightly better runtimes. We aim to investigate MPCitH signatures based on a combination of Rain and Limbo in future work.

2 Using Single-Key Even–Mansour

In this section we discuss using the single-key Even–Mansour (EM) construction as a one-way function. We start with background and security analysis, and give the construction we use for MPCitH-based signatures below in Section 2.1.

The single-key Even–Mansour scheme [DKS12, EM97, Riv84] is a way to construct a block cipher F from a cryptographic permutation π by adding a key k to the input x and to the output of the permutation, i.e.,

$$F_k(x) = k + \pi(x + k)\,. \tag{1}$$

In practice we can instantiate π with a block cipher such as AES, by fixing a random key p_0, and making it a public constant. We thus avoid calculating the non-linear key schedule with a secret key. We write $F : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, where n is the block size, and κ is the key size and $\pi : \{0,1\}^n \to \{0,1\}^n$. We assume that $\kappa = n$ and addition of n-bit strings is done with bitwise XOR.

Security. In the context of a signature scheme constructed from an MPC-in-the-head proof system, F must be a one-way function of the key: given some (x, y) such that $y = F_k(x)$, it should be difficult to find a k' such that $F_{k'}(x) = y$. More formally, F is a secure OWF if the following probability

$$\Pr[x \xleftarrow{\$} \{0,1\}^n, k \xleftarrow{\$} \{0,1\}^\kappa, y \leftarrow F_k(x), k' \leftarrow A^\pi(x,y) : F_{k'}(x) = y] \tag{2}$$

is negligible in n for all polynomial-time adversaries A, with oracle access to π.

It is easy to see that removing either addition of k in the construction of F is insecure. The single-key EM scheme is a special case of the two-key scheme, that constructs a block cipher T from π using two keys as $T(k_1, k_2, x) = k_2 + \pi(x + k_1)$. The function T is trivially not a secure one-way function (OWF) of the correct type: given (x, y) we can compute a (k_1, k_2) by first choosing k_1 at random and then setting $k_2 = y + \pi(x + k_1)$. It is therefore important to formally show security of the single-key EM construction for signatures.

We model π as an ideal permutation, and consider attacks where the attacker is given oracle access to it. We would like to prove a lower bound on the number of queries to π for a successful attack, or equivalently, show that any attack making q queries succeeds with probability negligible in κ. In [EM97] Even and Mansour prove a lower bound for the two-key construction, namely that an attack has $DT = \Omega(2^n)$ where D is the data complexity (i.e., the number of encryption queries or decryption queries to F) and T is the time complexity (measured as the number of queries to π). Dunkelman, Keller and Shamir [DKS12] show that with small modifications, the proof also holds for the single-key EM construction. However, the analysis of [EM97, DKS12] uses a reasonable but nonstandard CCA-like security definition. As shown above with the two-key construction, this definition does not imply security of EM in our setting. Dunkelman et al. comment that obtaining a nontrivial amount of information about the key is covered by the analysis, and our two-key attack above does not contradict this since the key produced by the attack is different from the original key used to create y with overwhelming probability. The main difference is that our OWF game may be won with *any* key for F while in the block cipher setting the attack must find the single correct key, in order to recover information about the plaintext.

Theorem 1. *The single-key Even–Mansour construction (Eq.* (1)*) is a secure one-way function, when the permutation π is an ideal random permutation.*

Proof. The attacker A is initialized with (x, y) and has oracle access to π. We must show that the probability in Eq. (2) is negligible in n (the key size and block size in bits).

We say that a key K is *consistent* with the pair (x, y) if $y = F_K(x)$, i.e., $y = K + \pi(x + K)$.

Just before producing an output, A has made q queries to π, and has q pairs (X_i, Y_i) where $Y_i = \pi(X_i)$. W.l.o.g., we assume that inputs X_i to π are distinct.

Therefore, each query X_i has the form $X_i = x + K_i$ for a distinct key K_i. In one case, the query either gives A the correct key, and we have $Y_i + K_i = y$. In the other case, the query does not give A the correct key, but reveals that $K'_i = y - Y_i$, a second key distinct from K_i, cannot be consistent with (x, y). First we note that K'_i is distinct from K_i since $Y_i + K_i \neq y$, then $K_i \neq y - Y_i$. To see that K'_i cannot be consistent if K_i is not consistent, assume that K'_i was consistent, which implies that

$$
\begin{aligned}
Y_i + K'_i &= \pi(x + K'_i) + K'_i \\
\pi(x + K_i) + K'_i &= \pi(x + K'_i) + K'_i \\
\pi(x + K_i) &= \pi(x + K'_i) \ ,
\end{aligned}
$$

which is a contradiction since π is a permutation and $K_i \neq K'_i$.

Now we argue that the remaining $2^n - 2q$ keys are equally likely to be consistent with (x, y), given the information A has. Consider K^*, one of the remaining keys, not equal to K_i or K'_i. If A knew that K^* was not consistent with (x, y), then A knows $y - K^*$, and that

$$y - K^* \neq \pi(x + K^*) \tag{3}$$

But since π is a random permutation, $\pi(x + K^*)$ is uniformly selected from the $2^n - 2q$ remaining possible values, so Eq. (3) holds with probability $1 - 1/(2^n - 2q)$ and all remaining keys are equally likely.

Therefore, after q queries A succeeds with probability not more than $2q/2^n$. Alternatively, A succeeds after $\Omega(2^n)$ queries.

$\square$

When compared to the upper bound given by the complexity of a brute-force attack (where A succeeds with probability $q/2^n$), this lower bound is off by a factor of two. As described in the proof, each query to π can be used to rule out two keys, improving the brute force attack, to succeed with probability $2q/2^n$. Therefore the lower bound is tight, since it matches the complexity of the attack.

The assumption required for security of the EM-OWF is that π is an ideal random permutation. Assume that π is constructed by fixing the key of a block cipher E (since this is how we will construct π for signature schemes). How this assumption compares to directly assuming E is a OWF is not always clear. In general, the ideal permutation assumption is stronger than directly assuming that E is an OWF of the key (as is currently done in signature schemes such as Picnic and Banquet) since a function can still be one-way even with slightly biased outputs, and we can construct contrived examples of OWFs with biased output. But is the difference meaningful for any natural choices of E?

When E is 5-round AES, Grassi, Rechberger and Rønjom describe a distinguisher [GRR17] that requires 2^{32} (plaintext, ciphertext) pairs and works for any key (for adaptively chosen ciphertexts improved to $2^{27.2}$ in [BR19b] and to 6 rounds in [BR19a]). In short, if the 2^{32} plaintexts are chosen carefully, then [GRR17] shows that the set of ciphertexts possess a property (with probability 1) that would not be present for an ideal permutation (except with low probability). Therefore, Theorem 1 says nothing about the security of the EM-OWF construction when π is built with 5-round AES.

By contrast, 5-round AES appears to be a secure OWF, as there are no known key-recovery attacks that work with a single (plaintext, ciphertext) pair. The best known attack in this class is given by Bouillaguet, Derbez and Fouque [BDF11], applied to 4-round AES and is marginal, costing 2^{120} time and 2^{80} memory.

In one sense there is more positive evidence that 5-round AES is a secure OWF than there is for EM-OWF, however, neither problem is very well studied, so making conclusions with confidence is difficult. The issue of how the security of the EM and non-EM one-way functions compare for practical choices of E (such as 5-round AES) is an interesting open question.

Multi-Target Security. We are also interested in the multi-target, or multi-user OWF security of the single-key EM construction. In the context of a signature scheme, each of ℓ users has a public key (x_i, y_i) and an attacker must find a key K such that $y_i = F_K(x_i)$ for *any* $i \in [\ell]$. Ideally the cost of attacking any one of ℓ users is at least as expensive as attacking one user.

There are some multi-target attacks known for single-key EM in the context of encryption. Fouque, Joux and Mavromati [FJM14] give a multi-target key recovery attack (that requires multiple (adaptive) queries to F, which are not possible when F is used as a key generation function for signatures). Mouha and

Luykx [ML15] prove that the advantage of distinguishing multiple EM instances from multiple permutations depends on the number of instances.

To mitigate multi-user attacks we consider two possible countermeasures. The first is to choose a random x_i per user, and the second is to choose a random permutation π_i per user. Without mitigation, a simple multi-target attack in our scenario is possible: simply guess K^*, compute $y^* = F_{K^*}(x)$ for the fixed x, and compare y^* to all y_i; if a match is found, then K^* is consistent with the matching (x, y_i). Another attack remains when x_i is random, but π is fixed. Note that a query X to π corresponds to some key for each user, namely K_i such that $X = K_i + x_i$. Then one key for each user can be tested by checking whether $K_i - X = y_i$, and so the queries to π are amortized across all users, and security loses a factor ℓ.

To create a random π_i each user also generates a random key as the public constant for the the block cipher used to construct π, and outputs the public key (x_i, y_i, π_i). Once π is random per user, it does not appear necessary to also choose random x_i, so we fix it to zero.

Post-Quantum Security of Single-Key EM. As with any one-way function, inputs can be recovered in time $O(2^{n/2})$ using quantum amplitude amplification [BHMT02] (a generalization of Grover's algorithm [Gro96]), with costs similar to those for AES as described in [JNRV20]. For OWF security, this appears to be the best known quantum attack. When sufficient queries may be made to F_k, attacks may become more efficient than key search [BHN$^+$19, HS18], but these are out of scope in the OWF context.

2.1 EM-OWF Constructions

Given a block cipher $E_k(x)$ with n-bit key size and block size, where k is the secret key and x is the input, we build the one-way function in k as

$$F(x, k) = k + E_x(k)\,. \tag{4}$$

As discussed above, this is an instance of the single-key EM construction when π is random per user, and the plaintext input (denoted x above) is fixed to zero. For example when E is AES-128 we have a suitable OWF for use in Banquet key generation at security level L1, here $n = 128$ bits. A requirement for Banquet and BBQ is that the key be chosen so that there are no zero inputs to S-boxes, and key generation uses rejection sampling to find one (each sample is rejected w.p. $\approx 1/2$). Note that with the EM construction, we can instead sample k uniformly at random, then vary x until $E_x(k)$ has no zero S-box inputs.

There is no direct way to use the EM-OWF construction at security levels L3 and L5, because the block size of AES is limited to 128 bits. These higher security levels are best achieved with Rijndael [DR98], where n can be 192 and 256 bits. Rijndael is not a standardized primitive (as AES-192 and AES-256 are), however it is a mature and well analyzed design.

For key generation in Picnic, since n scales to 192 and 256 in LowMC, all three security levels may use the construction of Eq. (4) directly. As the key schedule in LowMC consists of only linear operations, using the EM construction will not reduce signature sizes, but can improve performance and simplify implementation, since deriving round keys is done with a matrix multiplication.

3 LSAES: AES with Larger S-boxes

In [dDOS19] and [BdSGK+21] it has been shown that MPC protocols using the fact that the S-box of AES-128 is based on field inversion can reduce the size of AES-based signatures. In this section, we present a mild tweak to AES-128. We leave the description of AES-128 unchanged, except for the SubBytes part, where we replace the parallel byte-wise inversion in $\mathbb{F}_{2^8}$ of 4 elements in a row with a single inversion in $\mathbb{F}_{2^{32}}$. Essentially, this transforms AES-128 back to its predecessor having a 10-round SHARK-like [RDP+96] cipher with 32-bit S-boxes, which we call LargeSboxAES-128 or LSAES-128 in short.

It is possible to consider LSAES with 16-, 32-, 64- or 128-bit S-boxes (with minor changes to the key schedule). For use in Banquet at security level L1, 32 bits are a natural choice, because the MPC protocol already lifts the 8-bit S-box (input, output) pairs to a 32-bit field to increase the soundness of the polynomial checking protocol. We will therefore focus on 32-bit S-boxes. We briefly discuss the signature sizes achievable by using LSAES with other S-box sizes in Appendix D.2.

3.1 Specification

Since LSAES-128 corresponds largely to AES-128, we only recall the structure of the cipher and the application of the S-boxes in this section. The details regarding the affine parts and the key schedule can be found in Appendix B. As AES-128, LSAES-128 mainly consists of two parts, the key schedule, where the round-keys $k^{(i)}$ are derived from the secret key k, and the data path, where an input x is transformed by the round function R_i and mixed with the round-keys $k^{(i)}$. There are ten rounds and eleven round keys. Thus, we have

$$F_k(x) = R_{10} \circ \cdots \circ R_2 \circ R_1(x),$$

where each round function R_i $\forall i < 10$ is defined as

$$R_i(x) = L\left(S\left(x + k^{(i)}\right)\right) ,$$

and the last round function R_{10} is defined as

$$R_{10}(x) = S\left(x + k^{(10)}\right) + k^{(11)} .$$

3.1.1 Structure of the State

For describing the cipher, we use the typical description of AES-128 as a rectangular 4×4 representation of the bytes x_i, shown in Fig. 1. A 128-bit input x to the cipher is seen as a concatenation of byte elements $x = x_0 \mid\mid x_1 \mid\mid \cdots \mid\mid x_{15}$. In addition, we define the 32-bit elements $\chi_i = x_i \mid\mid x_{i+4} \mid\mid x_{i+8} \mid\mid x_{i+12} \quad \forall\, i \in \mathbb{N} : \quad 0 \leq i < 4$ leading to the representation shown in Fig. 2.

3.1.2 Substitution Layer S

The substitution layer interprets the four 32-bit state values χ_i as elements of the field GF(2^{32}) defined by the irreducible polynomial $X^{32} + X^7 + X^3 + X^2 + 1$ (a different choice of irreducible polynomial is possible in case of existing software

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

Figure 1: Byte-wise state.

χ_0
χ_1
χ_2
χ_3

Figure 2: 32-bit-wise state.

or hardware implementations). Then, the inversion operation is applied to each of the four χ_i with the additional convention that element $\mathbf{0}$ maps to $\mathbf{0}$. So we have

$$S(x) = S(\chi_0, \chi_1, \chi_2, \chi_3) = (\chi_0^{-1}, \chi_1^{-1}, \chi_2^{-1}, \chi_3^{-1}).$$

3.2 Cryptanalysis

One major feature of AES-128 is its resistance against differential [BS91] and linear [Mat94] cryptanalysis. Hence, those are not the dominant attack vectors covering the highest number of rounds. After years of cryptanalysis, attack vectors that can turn the strong alignment of AES's round function to their advantage, like impossible differential attacks [MDRMH10] and meet-in-the-middle [DFJ13] attacks, have turned out to cover the highest number of rounds. This section shows that LSAES retains excellent resistance against differential and linear cryptanalysis while also reaching at least the same security level against attacks exploiting strong alignment. Note that we consider the strongest attacks on AES without restricting the data complexity.

3.2.1 Differential and Linear Cryptanalysis

The use of 32-bit S-boxes in the columns of the AES state while leaving all linear transformation intact essentially leads to LSAES being a SHARK-like [RDP+96] design. The reasons for this are as follows (referring to the notation in Appendix B).

- The linear transformation AB_i is performed four times in parallel on 8-bit chunks of our 32-bit state variable values χ_i, while *SR* is just a re-ordering of bytes within each state value χ_i. Hence, we can see those two linear layers as part of 32-bit S-boxes, which are now just affine equivalents to the inversion in GF(2^{32}).
- Applying four 4×4 MDS matrices with coefficients in $\mathbf{F}_2^{8\times 8}$ in parallel on 8 bit parts of our state words χ_i corresponds to the application of a single 4×4 MDS matrix with coefficients in $\mathbf{F}_2^{32\times 32}$ [KLSW17].

Hence, we now know that a linear or differential trail has at least five active S-boxes per 2 rounds. So we have at least 25 active S-boxes in total. Our S-boxes have a maximum differential probability of $\delta = 2^{-30}$ and a maximum correlation of $\lambda = 2^{-15}$ [Nyb94]. Hence, the theoretical values for 10 round differential trails of $\delta = 2^{-750}$ and for 10 round linear trails of $\lambda = 2^{-375}$ let us conjecture that those attack vectors do not pose a threat to LSAES.

3.2.2 Attacks Exploiting Strong Alignment

AES has a strongly aligned structure that leads to nice bounds on the differential probability and correlation of trails, used to give convincing arguments that AES resists differential and linear cryptanalysis. If we take a closer look at one round of AES, we see that the only function directly providing diffusion between byte-borders is MixColumns. In particular, MixColumns is a linear function mixing the 4 bytes within each of the four columns of the AES state separately.

The attacks that reach the largest number of rounds for AES-128 exploit these characteristics of AES. Attacks that exploit the strong alignment of AES-128 are (amongst others) the Integral/Square attacks [DKR97, FKL$^+$01], impossible differential attacks [MDRMH10], or meet-in-the-middle attacks [DS08], where the last one gives the most efficient attacks on seven rounds of AES-128 [DFJ13].

For LSAES-128, the diffusion of the linear layer stays the same. The only thing we change is the S-box. Since the S-boxes of LSAES-128 are applied on all 4 bytes belonging to one row, we now have one additional component directly providing diffusion between byte-borders. In contrast to MixColumns, those S-boxes used in LSAES-128 are non-linear functions. Hence, we expect that all attacks exploiting the strong alignment and the direct linear mixture of the bytes in AES-128 do not perform better for LSAES-128. On the contrary, we expect that most of them will actually perform worse. For instance, we have implemented LSAES-128 in a tool capable of finding meet-in-the-middle and impossible differential attacks [DF16]. In contrast to AES-128, this tool does not give any indications for the existence of meet-in-the-middle and impossible differential attacks for seven rounds of LSAES-128.

3.2.3 Algebraic Attacks

Attempts to use the relatively simple algebraic structure for attacks on AES [CP02, MR02] turned out to be not fruitful [CL05, CMR06]. We conjecture that this also applies to LSAES.

4 Rain

Compared to LSAES, we go further with RAIN by increasing the S-box size and also modifying the linear layer. We use inversion in $\mathbb{F}_{2^n}$, where n is the state size. Compared to many other settings, the actual structure of the linear layer does not impact the performance much for our MPCitH use-cases. Hence, we use multiplication by a randomly chosen $n \times n$ matrix M over $\mathbb{F}_2$ as linear layer in order to improve the diffusion compared to more structured linear layers. Having such an unstructured linear layer is the main way that RAIN differs from JARVIS, which also uses an inversion in $\mathbb{F}_{2^n}$ as S-box.

Another way of seeing RAIN is as an SPN instance of the ideas proposed by Nyberg [Nyb94], or an iteration of a larger variant of AES's S-box [DR20] using a more unstructured affine mapping. Since, to the best of our knowledge, Nyberg [Nyb94] proposed the use of an inverse augmented affine permutation, we call our proposal RAIN (Random Affine Inverse Nyberg-inspired).

4.1 Specification

We define a keyed permutation $F_k(x)$ consisting of a nonlinear operation S and a constant addition over a large field $\mathbb{F}_{2^n}$, together with a linear layer. We choose $S : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ such that

$$S(x) = x^{2^n-2} = \begin{cases} x^{-1} & \text{if } x \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

The entire permutation is described as

$$F_k(x) = R_r \circ \cdots \circ R_2 \circ R_1(x),$$

where each round function $R_i \; \forall i < r$ is defined as

$$R_i(x) = M_i \left(S \left(x + k + c^{(i)} \right) \right),$$

and the last round function R_r is defined as

$$R_r(x) = k + S \left(x + k + c^{(r)} \right).$$

We denote the round constant in round i by $c^{(i)}$, and $M_i \in (\mathbb{F}_2)^{n \times n}$ is the linear layer matrix over $\mathbb{F}_2$ used in round i. Every such matrix M_i can also be represented as a linearized polynomial $L_i \in \mathbb{F}_{2^n}[X]$, namely

$$L_i(X) = \sum_{j=0}^{n-1} a_j^{(i)} X^{2^j}$$

for some coefficients $(a_j)_{j=0}^{n-1}$. In our design, we will make sure that $a_j \neq 0$ for $j \in \{0, \ldots, n-1\}$. This implies that the linearized polynomial is of maximum degree and as dense as possible.

A graphical overview of the construction is shown in Fig. 3. The details for the generation of the round constants and matrices are given in Appendix C.

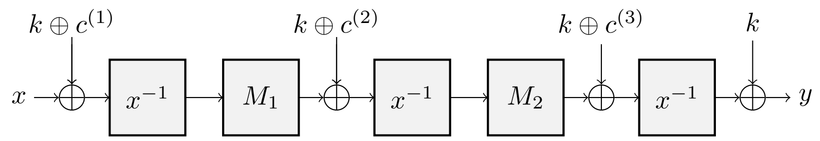


Figure 3: The Rain encryption function with $r = 3$ rounds. M_i denotes the multiplication with an unstructured invertible matrix over $\mathbb{F}_2$ in the i-th round.

4.2 Concrete Instances

- **128-Bit Security.** For our instance having 128 bits of security, we use the same irreducible polynomial as AES-GCM [MV04]. Hence, we have the field $GF(2^{128})$ defined by the irreducible polynomial $X^{128} + X^7 + X^2 + X + 1$.
- **192-Bit Security.** For a 192-bit security, we use the field $GF(2^{192})$ defined by a pentanomial $X^{192} + X^7 + X^2 + X + 1$.
- **256-Bit Security.** For a 256-bit security we use the field $GF(2^{256})$ defined by a pentanomial $X^{256} + X^{10} + X^5 + X^2 + 1$.

4.3 Design Rationale

The signature sizes of Banquet [BdSGK+21] benefit from using symmetric encryption schemes, which use only a few finite field inversion as non-linear operation. Simultaneously, the choice of the affine mixing layer does not play a significant role in signature sizes and only mildly influences the runtime. Hence, we followed the natural path provided by these insights using a cipher that uses few large inversions combined with a very strong affine diffusion layer consisting of unstructured binary matrices that also have a dense linearized polynomial that is of maximum degree when expressed over the same field as the inversion.

While these choices of components make sense in the context of MPCitH-based signatures, the resulting cipher does not perform well in classical applications since neither large finite field inversions nor unstructured binary matrix multiplication can be implemented very efficiently. Recently designed ciphers (such as *Vision* [AAB+20] and Jarvis [AD18]) that use large finite field inversions use an affine layer that can be efficiently described in the same field. In comparison, the very unstructured affine layer of Rain combined with the limit in the data complexity implied by the used signature scheme allows us to reduce the number of field inversion further, while aiming at the same level of security.

Since we use Rain with only a small amount of rounds, we have decided to use different matrices and round constants per round to improve the diffusion as much as possible and prohibit symmetry properties. The use of different round constants and matrices per round only has a negligible effect on the performance compared to always using the same matrix in each round and no round constants. Furthermore, we benefit from using block ciphers in an Even-Mansour mode in this paper because the key schedule does not have to be computed. We take this observation into consideration when designing the key schedule for Rain. Hence, we decided to use the trivial key schedule by defining $k_i = k$ to maximize the performance of Rain.

4.4 Cryptanalysis

In this section, we will show that 3 rounds of the construction are sufficient in order to provide security. Indeed, we do not consider fewer than 3 rounds, mainly to avoid the possibility of splitting the construction into two single-round parts.

For the sake of simplicity, here we omit using the round-specific indices of the linearized permutation polynomials L_i, i.e., we write L instead of L_i. Further, in our scenario the attacker is only able to use a single (plaintext, ciphertext) pair (p, c), and the goal is to recover a full key k which maps c to p. This is due to its specific use in the signature scenario, where the plaintext/ciphertext pair is fixed in the public key and does not change, only ever giving an attacker access to a single, randomly chosen plaintext/ciphertext pair.

4.4.1 Gröbner Basis Attacks

We try to represent the keyed permutation as a system of equations, which we then want to solve by converting it into a Gröbner basis and solving the univariate equations (after order transformation) for the remaining variables.

Equation System over $\mathbb{F}_2$. A straightforward way to build such a system is by working with variables in $\mathbb{F}_2$. We would then have n variables and n equations for a key size of n bits. Each equation describes a single ciphertext bit in the (known) plaintext bits and the key variables. Working purely with Gröbner bases, we assume this system of equations to be hard to solve, as each round has an algebraic (bit-level) degree of $n-1$, which is the maximum for a permutation.[1] Note that this is true also for the decryption direction, since we use the inverse function. Hence, we conjecture that 3 rounds of this permutation are sufficient in order to provide security w.r.t. Gröbner basis attacks using equation systems over $\mathbb{F}_2$.

Equation System over $\mathbb{F}_{2^n}$ (Single Variable). A more efficient approach is to instead focus on equation systems over $\mathbb{F}_{2^n}$. As we only have a single input then, we can represent the whole equation by a single equation in a single key variable (we assume a linear key schedule). Note that the resulting system of equations with a single equation is automatically a Gröbner basis, hence this step is free. However, we still need to account for the complexity of solving this equation for the single key variable. Indeed, with overwhelming probability, the resulting polynomial has a maximum degree of $2^n - 2 = d$ over $\mathbb{F}_{2^n}$ and is also dense. The cost of finding the root of such a polynomial is an element in $\mathcal{O}(d^3n^2 + dn^3)$ [Gen07]. Since the maximum degree is already reached after a single round, we conjecture that 3 rounds provide ample security w.r.t. to this approach.

Equation System over $\mathbb{F}_{2^n}$ (Intermediate Variables). Another strategy is to add intermediate variables in order to decrease the degree of the inverse operation. This can be done by exploiting the fact that

$$x^{-1} = y \implies xy = 1$$

for every nonzero $x \in \mathbb{F}_{2^n}$. Hence, instead of using a single equation for the whole permutation, we can introduce an intermediate variable in each round and use it to translate every nonlinear operation into a degree-2 relation. This fact holds with a high probability of $(1-2^{-n})^r$ when using r rounds. Then, the equation system for r rounds consists of r variables and r equations in $\mathbb{F}_{2^n}[x_1, x_2, \ldots, x_{r-1}, k]$, namely

$$\begin{aligned} \left(p + k + c^{(1)}\right) \cdot L^{-1}(x_1) - 1 &= 0, \\ \left(x_1 + k + c^{(2)}\right) \cdot L^{-1}(x_2) - 1 &= 0, \\ &\vdots \\ \left(x_{r-2} + k + c^{(r-1)}\right) \cdot L^{-1}(x_{r-1}) - 1 &= 0, \\ \left(x_{r-1} + k + c^{(r)}\right) \cdot (c - k) - 1 &= 0. \end{aligned} \tag{5}$$

Note that the last equation has a degree of 2, while the other $r-1$ equations have a high degree, since we assume L^{-1} to be a linearized polynomial of high

[1] Indeed, current results in the literature suggest that even low-degree equation systems over $\mathbb{F}_2$ are hard to solve (see e.g. [JV17, NNY18]).

degree.[2] Concretely, using this system of equations and generic estimates, we would assume the degree of regularity of this system to be

$$D_{\text{reg}} = 1 + \sum_{i=1}^{r} \deg(f_i) - 1 = 3 + \sum_{i=1}^{r-1} 2^{n-1} - 1 = 3 + (r-1)(2^{n-1}-1) \geq 2^{n-1} \quad (6)$$

for $r \geq 2$ and practical n. We immediately see that

$$\left(\binom{r+1+2^{n-1}}{2^{n-1}}^{\omega}\right) > \left(\binom{1+2^{n-1}}{2^{n-1}}^{\omega}\right) = (1+2^{n-1})^{\omega} > 2^{n}$$

and hence

$$\log_2\left(\binom{r+1+2^{n-1}}{2^{n-1}}^{\omega}\right) > n$$

for $\omega \geq 2$ and $r \geq 1$. However, note that this analysis assumes that the equations result from a semi-regular system of equations, which is clearly not the case due to the linearized (and hence sparse) polynomial L. Therefore, we evaluated the actual degree reached in practical experiments.

Degrees Reached During the Computation. An overview of some of our results is given in Fig. 4. In this illustration, D_{reg} denotes the estimated degree of regularity assuming a semi-regular system of equations. On the other hand, D denotes the highest degree reached during a Gröbner basis computation.

In our tests, we observed that the minimum degree resulting from one possible reduction between two high-degree equations (i.e., a degree of $2 \cdot 2^{n-1} = 2^n$) is always reached for $r \geq 3$.[3] Indeed, the actual degree reached in experiments matched the estimated degree for $r = 3$ and came very close for $r = 2$. In many cases, D was also equal to the *first-fall degree*.

Further, the complexity of the Gröbner basis conversion using e.g. the FGLM algorithm [FGLM93] depends on the degree of the final recovered univariate polynomial in the lexicographically ordered basis. In our tests, this degree reached its maximum for $r \geq 3$, and the resulting polynomial was also dense. Following these results, we conjecture that a similar behavior can also be observed for larger block sizes and that the construction provides security if $r \geq 3$.

Toy Versions and Comparison with Exhaustive Search. The Gröbner basis computations for block sizes of $n > 10$ bits start to get increasingly expensive, mainly due to the high degrees in the computation. For these versions with reduced block sizes, we can confirm that the attacks perform significantly worse than a simple exhaustive search on a small number of bits. We expect that the same is also true for practical block sizes.

A Different Representation. The two representations given above (i.e., one variable for the entire permutation or one intermediate variable for each round)

[2]Indeed, we generate L^{-1} such that it is guaranteed to have a high degree.

[3]Since we have $r-1$ linear layers, we only have a single linear layer for $r = 2$ rounds, and hence only one high-degree equation in the resulting equation system.

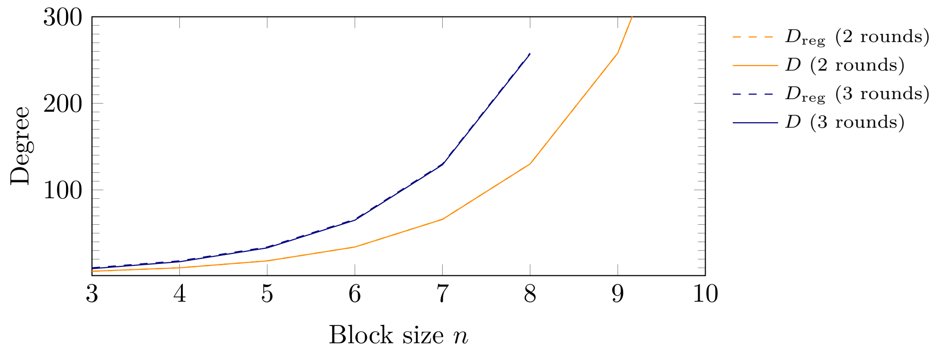


Figure 4: Comparison of the estimated degree of regularity and the highest degree encountered in the Gröbner basis computation when considering $r \in \{2, 3\}$ rounds. D_{reg} was computed using Eq. (6) and D is the highest degree reached in practical experiments. We note that some of the lines are overlapping and thus cannot be seen clearly, in which case the estimated degree matches the practical one.

are not the only possible descriptions of the function. Indeed, given Eq. (5), for example it is possible to skip some variables and equations, since

$$L^{-1}(x_i) = \frac{1}{x_{i-1} + k + c^{(i)}} \implies x_i = L\left(\frac{1}{x_{i-1} + k + c^{(i)}}\right), \text{ and}$$
$$x_i + k + c^{(i+1)} = \frac{1}{c - k} \implies x_i = \frac{1}{c-k} - k - c^{(i+1)},$$

and hence

$$L\left(\frac{1}{x_{i-1} + k + c^{(i)}}\right) = \frac{1}{c-k} - k - c^{(i+1)},$$

which implies

$$\left(a_{n-1} + \sum_{j=0}^{n-2} a_j \left(x_{i-1} + k + c^{(i)}\right)^{2^{n-1} - 2^j}\right)(c - k)$$
$$- \left(x_{i-1} + k + c^{(i)}\right)^{2^{n-1}} \left(\left(k + c^{(i+1)}\right)(c - k) - 1\right) = 0$$

for the last round, where $(a_j)_{j=0}^{n-1}$ are the coefficients of L (we omit the distinction between the different linear layers for simplicity). A similar technique can also be used to skip every second variable in a construction using more than 3 rounds. However, even though this approach may speed up the Gröbner basis computations for a low number of equations and variables (although the degree of regularity was the same in practical experiments), this does not necessarily mean that the final solving step will also be faster. In particular, in our practical tests we observed that the recovered univariate polynomials were mostly dense and of maximum degree, especially with larger n. Exploiting this approach further, we would eventually end up at the single-variable equation system given at the beginning of this section, where the prohibitively expensive step is not the Gröbner basis computation, but the factorization of the final polynomial.

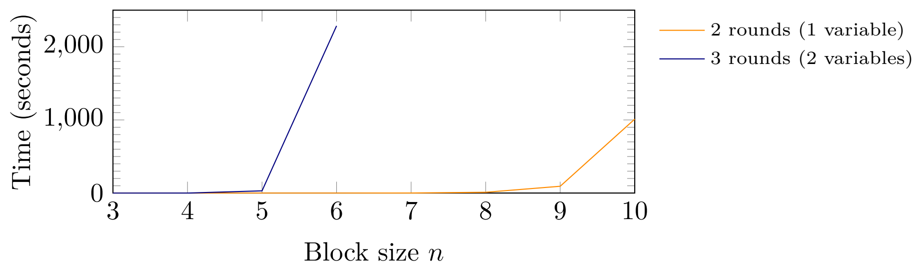


Figure 5: Solving time comparisons for 2-round and 3-round versions of Rain.

Concrete Runtime Results. In Fig. 5 we compare the runtime of the entire solving step for a key-recovery attack and different block sizes n. Our experiments suggest that the expected runtime more than doubles when increasing n by one for $n \geq 6$. We therefore reasonably assume that the same behavior continues for larger block sizes, and in particular for $n \geq 128$. All tests were done with `Sage`.

Comparison with Similar Constructions. Note that this design is similar to Jarvis [AD18], since it uses the same S-box and a similar round function. Moreover, there have been attacks on Jarvis [ACG+19] which use a similar approach as the one taken here. However, these attacks mainly exploit the low degree of the linear layer (which is required by the design strategy). This is not possible for Rain, since we ensure that each linear layer has maximum degree. Indeed, in [ACG+19, Section 7], the authors compare the affine part of the Jarvis S-box with the affine part of the AES S-box. The latter also has a maximum degree and cannot be split into two low-degree components, which is possible in Jarvis.

Moreover, our construction shows similarities with MiMC [AGR+16], whose S-box also covers the full state and can also be described by low-degree polynomials. Although MiMC has recently been attacked [EGL+20], the proposed method is not applicable in our setting.

4.4.2 Other Attacks

Since we only consider an attacker who has access to a single (plaintext, ciphertext) pair, we mainly analyzed attack vectors which are based on solving equation systems over some finite field. However, for completeness, we briefly mention other attack vectors here.

Differential and Linear Attacks. Note that these statistical attacks in their basic form need multiple (plaintext, ciphertext) pairs and are thus not directly applicable in our scenario. Still, even when considering multiple pairs, we would be counting the number of active S-boxes in each round. Since our design consists of a single large S-box, for a nonzero input difference in each round clearly one S-box is active. Further, the inversion mapping over $\mathbb{F}_{2^n}$ provides sufficiently strong linear and differential properties [Nyb94] such that the maximal differential probability is less than 2^{-2n} and the maximum linear correlation is less than 2^{-n} after 3 rounds of our construction.

Interpolation. We do not consider interpolation attacks in our setting, since multiple data pairs are needed in order to interpolate. However, we still assume the polynomial representation of the function to be dense after 3 rounds, mainly due to the uses of the inversion mapping.

Higher-Order Differential Attacks. In higher-order differential attacks [Lai94], a potentially low degree of the construction over $\mathbb{F}_2^n$ (i.e., the *algebraic degree*) is exploited. However, this attack vector also needs more than a single pair, and is hence not directly applicable in our setting. Nevertheless, variations of the attack might work for very low degrees [DMRS20]. Therefore, we mention that the algebraic degree of the inversion mapping in $\mathbb{F}_{2^n}$ is $n-1$ (i.e., the maximum for a permutation), and hence a higher-order differential distinguisher is infeasible since it would use the full space. The same is true also for the inverse direction.

4.4.3 Recommended Number of Rounds

From the security analysis just given, we conclude that 3 rounds provide security against attackers who are only allowed to use a single (plaintext, ciphertext) pair. In the following, we primarily use 3 or 4 rounds and call the resulting instances $\textsc{Rain}_3$ and $\textsc{Rain}_4$, respectively. While our analysis shows that 3 rounds are sufficient, we propose to primarily use $\textsc{Rain}_4$, with an additional round of security margin. In our evaluation in Section 6, we give numbers for both the 3-round and 4-round variants for comparison. These round numbers are valid independent of the block size n.

5 Constructing Signatures from our Designs

We now discuss how one can use the designs presented in the previous sections to build post-quantum signature schemes.

5.1 EM-AES, LSAES, EM-LSAES

For the constructions described in Section 2.1 and Section 3, we use the Banquet proof system [BdSGK+21] directly. We give the parameters (in the notation of [BdSGK+21, Section 4]) for the different instantiations in Table 1. The number of parties N and parallel repetitions τ can be chosen as a tradeoff between size and speed, following the soundness analysis of [BdSGK+21, Section 6.1].

Key generation also follows Banquet and BBQ: secret keys are randomly sampled until one is found such that none of the S-box inputs are zero. As explained in Section 2.1 for the EM options we can instead sample the key uniformly at random, and choose the per-user random value until no S-box inputs are zero.

In the case of LSAES, the probability of a zero input per S-box is decreased from $1/256$ to $1/2^{32}$, and the total number of S-boxes is nearly four times lower, so the probability of a zero input per key, given by $(1-2^{-32})^{\#\text{ S-boxes}}$, is about 2^{-26}. As argued in [dDOS19, BdSGK+21], this excludes only a small portion of the key space, and does not reduce security significantly.

OWF	κ	m	m_1	m_2	n	λ
EM-AES-128	128	160	10	16	8	4
LSAES-128	128	50	5	10	32	1
EM-LSAES-128	128	40	5	8	32	1

Table 1: Parameters for the Banquet proof system [BdSGK+21] for κ-bit security, a total of m ($= m_1 \cdot m_2$) inversions over a native field of $\mathbb{F}_{2^n}$, which get lifted to an extension field $\mathbb{F}_{2^{n\lambda}}$ for the proof.

5.2 Rainier – A Signature Scheme based on Rain

For building a signature scheme from our Rain design, we do not use Banquet directly. Instead, we use a simplified variant of Banquet that is better suited for the low number of inversions in Rain. We give a short summary of Banquet and the modifications we make to the protocol in the following.

5.2.1 High-Level Roadmap

We rely on the ideas of Banquet [BdSGK+21], which introduced a new MPCitH proof protocol suited for block ciphers using field inverses. Instead of actually computing the inverse operation using a square-and-multiply approach or relying on other techniques from the MPC literature like the masked inversion of BBQ [dDOS19], the Banquet proof protocol instead injects for every S-box input s_i, the output $t_i = {s_i}^{-1}$ as an additional input to the protocol. While this removes the need to compute it during the MPC evaluation, the parties must validate that the injected values (s_i, t_i) are valid inverses.

A simple approach is to multiply s_i and t_i in the MPC protocol and check if the result is 1 (inputs $s_i = 0$ are filtered out during key generation). However, this again introduces additional overhead to perform the multiplications. An optimization proposed in Banquet is to instead interpolate two degree-$(m-1)$ polynomials S and T using all of the inputs and outputs of the m inverse operations, respectively, and finally show that, when multiplied together, they are equal to a degree-$(2m-2)$ polynomial P which evaluates to 1 at the points $[1, m]$. This check is done by evaluating the polynomials at a randomly chosen point R and checking that $S(R) \cdot T(R) = P(R)$. To make this polynomial proof zero-knowledge, an additional random point is included when interpolating S and T in the MPC protocol, making them degree-m polynomials and P a degree-$2m$ polynomial. This ensures that the evaluation of the polynomials at the point R does not leak any information about the interpolated values.

To further improve efficiency, Banquet first splits the m values into a square of size $\sqrt{m} \times \sqrt{m}$ and then interpolates the rows to get $\sqrt{m}$ polynomials of degree $\sqrt{m}$. Then, a random linear combination of these smaller polynomials is checked in a similar fashion as above. Compared to the above method, this reduces the number of elements that need to be communicated to other parties from $2m$ to $m + \mathcal{O}(\sqrt{m})$. However, in our investigation we observed that while this step is beneficial for Banquet where m is in the hundreds, for our Rain design, $m \in \{3, 4\}$. For such a low number, the additional overhead of this approach results in larger signatures. We therefore take a step back and use the simple polynomial checking protocol outlined above for Rain. In general, the crossover point seems to be for m around 40–50 and also depends on N and the field size. For instance, with EM-LSAES-128 (which uses a 32-bit field), we have $m = 40$, and we estimate that when $N \geq 256$ the simplified protocol produces shorter proofs.

In addition to producing smaller signatures (at least for Rain) and having a simpler protocol description, the simple polynomial checking protocol, which we call Rainier in the following, also transforms the proof from a 7-pass protocol to a 5-pass protocol. This means that when we make it non-interactive using the Fiat-Shamir transformation, we can rely on the previous analysis of 5-pass protocols by Kales and Zaverucha [KZ20a] for the soundness calculation. The new soundness analysis also allows us to reduce the number of parallel repetitions, further reducing signature size.

Further Optimizations. Notice that we do not need to include all 3 points $S(R)$, $T(R)$ and $P(R)$ in the final signature, since from any two of them we can calculate the third one. Therefore, we only include $S(R)$ and $T(R)$ in the final signature and save one field element per repetition. (Note that this can also be applied to the original Banquet protocol for minor size reductions.)

Additionally, in a similar fashion to Picnic, we can omit the inclusion of $\Delta\mathsf{sk}$, Δt and ΔP in the signature in case that the challenged party is the first one $(j = 1)$. This would save, on average, $1/N \cdot \tau \cdot (\kappa + L \cdot \kappa + (L + 1) \cdot \kappa)$ bits in the signature. However, for a larger number of parties this does not result in signigficant savings (e.g., for 128-bit security and 64 parties we save 44 bytes on average), and we therefore choose not to use this optimization, which has the drawback of making signatures variable-length. However, if future work were to use $N = 2$, this would be a significant optimization.

Remark 2. As mentioned, one alternative to the Banquet-style protocol above is to directly calculate $s_i \cdot t_i$ in the MPC protocol and check that the result is equal to one. This can be done using a variant of [KKW18] working over the native field. However, a quick analysis of the resulting signature size shows that this approach is inferior: A signature using this approach with Rain at the 128-bit security level has 512 bits of MPC input per repetition, 384 bits of preprocessing information and 384 bits of online communication, totaling 1280 bits per repetition. Together with the commitments and seeds, a KKW-style signature would be 12.9 KB in size, which is just a little larger than Picnic3 for the same parameter choices. As we will show in Section 6, Rainier can produce much smaller signatures, highlighting the interplay between the cipher and the MPCitH proof system.

5.2.2 The Rainier signature scheme

The key generation function of Rainier, $\mathsf{KeyGen}(1^\kappa)$ samples $k, p \overset{\$}{\leftarrow} \{0,1\}^\kappa$, computes $c \leftarrow \text{Rain}_k(p)$. If any of the inverse operations in the computation have a zero input, repeat with a new random p until there are no zero inputs. Output the secret key $\mathsf{sk} = k$ and the public key $\mathsf{pk} = (p, c)$. Due to the large size of the field, the probability of the zero case occurring is negligible.

We give the full signing algorithm of Rainier in Fig. 6 and Fig. 7, and show the verification algorithm in Fig. 8. These algorithms make use of several hash functions: Commit, H_1 and H_2; as well as two pseudorandom generators: Expand

and $\mathsf{ExpandTape}$. All of these are instantiated using SHAKE128[4] (or SHAKE256 for larger security levels), with different constants added for domain separation. The $\mathsf{Sample}(t)$ function is a helper function that samples elements from a random tape t that was output by $\mathsf{ExpandTape}$, remembering and advancing the current position on the tape.

5.2.3 Soundness Analysis and Parameter Selection

Rainier is parameterizable on the security level κ (which is also equal to the key and block size of our block cipher, and inverse operations are done in $\mathrm{GF}(2^\kappa)$), the number of parties N and the number of parallel repetitions τ.

In the following we give an analysis of the soundness of the protocol, based on the previous analysis done in [BdSGK+21] and taking into account our modifications to the protocol, like the reduction from 7 to 5 rounds. In Phase 2 of Fig. 7, a random value R_e is produced for each repetition which is the point at which the polynomials S_e, T_e and P_e are evaluated. Remember that for all $l \in [L]$ the values of $S_e(l)$ and $T_e(l)$ correspond to the input and output of the inverse operation respectively and that we set up P_e in such a way that $P_e(l) = 1$. If $S_e \cdot T_e \neq P_e$, we can bound the probability of a random challenge R_e not detecting this inequality by the number of zeroes of the polynomial $Q_e = P_e - S_e \cdot T_e$. In the case of a cheating prover, $Q_e(R_e)$ may be zero by chance, but the number of zeroes of Q_e is bounded by Lemma 3.

Lemma 3 (Schwartz–Zippel Lemma)**.** *Let $Q(x) \in \mathbb{F}[x]$ be a non-zero polynomial of degree $d \geq 0$. For any finite subset S of $\mathbb{F}$,*

$$\Pr\left[r \xleftarrow{\$} S : Q(r) = 0\right] \leq \frac{d}{|S|}\,.$$

Since the polynomial Q_e is of degree $2L$, and we sample the challenge R_e from a set of size $2^\kappa - 2L$, we arrive at a maximum cheating probability of $\frac{2L}{2^\kappa - 2L}$.

The second challenge in Phase 4 of Fig. 7 chooses the player whose state does not get revealed. A cheating prover has a $1/N$ chance of guessing this challenge and therefore cheating.

Following the analysis of 5-pass protocols without early abort in [KZ20a], we provide an attack strategy with the minimum work for an attacker, where we cheat τ_1 times for the first challenge (i.e., produce a polynomial $P_e \neq S_e \cdot T_e$ and hope that the random challenge hits a random zero in Q_e) and cheat in the remaining $\tau_2 = \tau - \tau_1$ instances by guessing which player does not get revealed (the second challenge) and cheat in the MPC computation of that player.

The minimum cost of the attack is then given by

$$\mathsf{Cost}(\kappa, N, \tau) = \frac{1}{\mathsf{SPMF}(\tau, \tau_1, 2L/(2^\kappa - 2L))} + N^{\tau - \tau_1}\,,$$

where SPMF is the summed probability mass function,

$$\mathsf{SPMF}(n, k, p) = \sum_{k'=k}^{n} \binom{n}{k'} p^{k'} (1-p)^{n-k'}\,,$$

[4]We note that hashing is the dominant part of the overall runtime and a faster hash function (e.g., SHA-2 or Haraka [KLMR16]) can substantially improve performance.

$\mathsf{Sign}(\mathsf{sk}, \mathsf{msg})$: **Phase 1: Committing to the seeds, the execution views and interpolated polynomials of the parties.**

Sample a random salt $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{2\kappa}$.
for each parallel execution e **do**
  Sample a root seed: $\mathsf{seed}_e \xleftarrow{\$} \{0,1\}^{\kappa}$.
  Derive $\mathsf{seed}_e^{(1)}, \dots, \mathsf{seed}_e^{(N)}$ as leaves of binary tree from seed_e.
  for each party i **do**
    Commit to seed: $\mathsf{com}_e^{(i)} \leftarrow \mathsf{Commit}(\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)})$.
    Expand random tape: $\mathsf{tape}_e^{(i)} \leftarrow \mathsf{ExpandTape}(\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)})$
    Sample witness share: $\mathsf{sk}_e^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$.
  Compute witness offset: $\Delta\mathsf{sk}_e \leftarrow \mathsf{sk} - \sum_i \mathsf{sk}_e^{(i)}$.
  Adjust first share: $\mathsf{sk}_e^{(1)} \leftarrow \mathsf{sk}_e^{(1)} + \Delta\mathsf{sk}_e$.
  for each S-box ℓ **do**
    For each party i, compute the local linear operations in Rain to obtain the share $s_{e,\ell}^{(i)}$ of the S-box input $s_{e,\ell}$.
    Compute the S-box output: $t_{e,\ell} = \left(\sum_i s_{e,\ell}^{(i)}\right)^{-1}$.
    For each party i, set their share: $t_{e,\ell}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$.
    Compute output offset: $\Delta t_{e,\ell} = t_{e,\ell} - \sum_i t_{e,\ell}^{(i)}$.
    Adjust first share: $t_{e,\ell}^{(1)} \leftarrow t_{e,\ell}^{(1)} + \Delta t_{e,\ell}$.
  Broadcast each party's share $\mathsf{ct}_e^{(i)}$ of the output.
  for each party i **do**
    Sample additional random points: $\bar{s}_e^{(i)}, \bar{t}_e^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$.
    Define $S_e^{(i)}(k) = s_{e,k}^{(i)}$ and $T_e^{(i)}(k) = t_{e,k}^{(i)}$ for $k \in [0, L-1]$ as well as $S_e^{(i)}(L) = \bar{s}_e^{(i)}$ and $T_e^{(i)}(L) = \bar{t}_e^{(i)}$.
    Interpolate polynomials $S_e^{(i)}(\cdot)$ and $T_{e,j}^{(i)}(\cdot)$ of degree L using the defined $L+1$ points.
  Compute product polynomial: $P_e \leftarrow \left(\sum_i S_e^{(i)}\right) \cdot \left(\sum_i T_e^{(i)}\right)$.
  for each party i **do**
    For $k \in [0, L-1]$: $P_e^{(i)}(k) = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{if } i \neq 1 \end{cases}$
    For $k \in [L, 2L]$, sample $P_e^{(i)}(k) \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$.
  for $k \in [L, 2L]$ **do**
    Compute offset: $\Delta P_e(k) = P_e(k) - \sum_i P_e^{(i)}(k)$.
    Adjust first share: $P_e^{(1)}(k) \leftarrow P_e^{(1)}(k) + \Delta P_e(k)$.
  For each party i, interpolate $P_e^{(i)}$ using the defined $2L+1$ points.
Set $\sigma_1 \leftarrow (\mathsf{salt}, ((\mathsf{com}_e^{(i)})_{i\in[N]}, (\mathsf{ct}_e^{(i)})_{i\in[N]}, \Delta\mathsf{sk}_e, (\Delta t_{e,\ell})_{\ell\in[L]},$
   $(\Delta P_e(k))_{k\in[L,2L]})_{e\in[\tau]}$.

Figure 6: Rainier Signature Scheme - Phase 1. Commitment to executions of Rain and the interpolated polynomials. We use e to index the τ parallel repetitions, i to index the N parties, and ℓ to index the L S-boxes.

where each term gives the probability of guessing correctly in k' of τ independent

Phase 2: Challenging the checking polynomials.

1: Compute challenge hash: $h_1 \leftarrow H_1(\mathsf{msg}, \mathsf{pk}, \sigma_1)$.
2: Expand hash: $(R_e)_{e\in[\tau]} \leftarrow \mathsf{Expand}(h_1)$ where $R_e \in \mathbb{F}_{2^\kappa} \setminus [0, L-1]$.

Phase 3: Committing to the views of the checking protocol.

1: **for** each execution e **do**
2:     **for** each party i **do**
3:         Compute and open: $a_e^{(i)} \leftarrow S_e^{(i)}(R_e)$ and $b_e^{(i)} \leftarrow T_e^{(i)}(R_e)$.
4:         Compute and open: $c_e^{(i)} \leftarrow P_e^{(i)}(R_e)$.
5: Set $\sigma_2 \leftarrow (S_e(R_e), T_e(R_e), P_e(R_e), (a_e^{(i)}, b_e^{(i)}, c_e^{(i)})_{i\in[N]})_{e\in[\tau]}$.

Phase 4: Challenging the views of the checking protocol.

1: Compute challenge hash: $h_2 \leftarrow H_2(\mathsf{salt}, h_1, \sigma_2)$.
2: Expand hash: $(\bar{i}_e)_{e\in[\tau]} \leftarrow \mathsf{Expand}(h_2)$ where $\bar{i}_e \in [N]$.

Phase 5: Opening the views of the checking protocol.

1: **for** each execution e **do**
2:     $\mathsf{seeds}_e \leftarrow \{\log_2(N)$ nodes needed to compute $\mathsf{seed}_{e,i}$ for $i \in [N] \setminus \{\bar{i}_e\}\}$.
3: Output $\sigma \leftarrow (\mathsf{salt}, h_1, h_2, (\mathsf{seeds}_e, \mathsf{com}_e^{(\bar{i}_e)}, \Delta\mathsf{sk}_e, (\Delta t_{e,\ell})_{\ell\in[L]},$
$(\Delta P_e(k))_{k\in[L,2L]}, S_e(R_e), T_e(R_e))_{e\in[\tau]})$.

Figure 7: Rainier Signature Scheme - Phases 2-5. Computation of the checking protocol, challenging and opening of the views of the checking protocol.

trials, each with success probability p. The choice of τ_1 that minimizes the attack cost gives the optimal attack

$$\tau_1 = \underset{0\le\tau'\le\tau}{\arg\min} \frac{1}{\mathsf{SPMF}(\tau, \tau', 2L/(2^\kappa - 2L))} + N^{\tau-\tau'} .$$

To select secure parameters, we fix κ and N and increase the value of τ until the best attack strategy has an average cost of 2^κ or more. A script to select secure parameters is included in the source code and was used to generate all parameter sets in the following section.

5.2.4 Security Proof

Although our protocol is similar in nature to the proof protocol of Banquet [BdSGK+21], the modifications to the protocol (reducing it from 7 to 5 internal rounds) mean that the existing proof does not apply directly. Like Banquet, we conjecture that Rainier is also secure in the quantum random oracle model (QROM), as there has been much recent progress in QROM analysis for signature schemes constructed from Σ-protocols (see, e.g., [DFMS19, DFM20, GHHM20]), but we leave a formal proof to future work. A good starting point is the work of Don et al. [DFM20], who give a generic QROM security analysis of signatures schemes constructed from 5-round Σ-protocols using the Fiat-Shamir transform.

Theorem 4. *The Rainier signature scheme is* EUF-CMA*-secure, assuming that* Commit*,* H_1*,* H_2 *and* Expand *are modelled as random oracles,* ExpandTape *is a secure PRG, the seed tree construction is computationally hiding, the* (N, τ, L) *parameters are appropriately chosen, and that* KeyGen *is a secure one-way function.*

$\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \sigma)$:

Parse $\sigma \leftarrow (\mathsf{salt}, h_1, h_2, (\mathsf{seeds}_e, \mathsf{com}_e^{(\bar{i}_e)}, \Delta\mathsf{sk}_e, (\Delta t_{e,\ell})_{\ell\in[L]}, (\Delta P_e(k))_{k\in[L,2L]}, S_e(R_e), T_e(R_e))_{e\in[\tau]})$.

Expand hashes as $(R_e)_{e\in[\tau]} \leftarrow \mathsf{Expand}(h_2')$ and $(\bar{i}_e)_{e\in[\tau]} \leftarrow \mathsf{Expand}(h_3)$.

for each execution e **do**

  Use seeds_e to recompute $\mathsf{seed}_e^{(i)}$ for $i \in [N] \setminus \bar{i}_e$.

  for each party $i \in [N] \setminus \bar{i}_e$ **do**

    Recompute $\mathsf{com}_e^{(i)} \leftarrow \mathsf{Commit}(\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)})$, $\mathsf{tape}_e^{(i)} \leftarrow \mathsf{ExpandTape}(\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)})$ and $\mathsf{sk}_e^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$.

    if $i \stackrel{?}{=} 1$ **then**

      Adjust first share: $\mathsf{sk}_e^{(i)} \leftarrow \mathsf{sk}_e^{(i)} + \Delta\mathsf{sk}_e$.

    for each S-box ℓ **do**

      Compute local linear operations in RAIN to obtain $s_{e,\ell}^{(i)}$.

      Sample output share: $t_{e,\ell}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$.

      if $i \stackrel{?}{=} 1$ **then**

        Adjust first share: $t_{e,\ell}^{(i)} \leftarrow t_{e,\ell}^{(i)} + \Delta t_{e,\ell}$.

    Recompute output broadcast $\mathsf{ct}_e^{(i)}$.

    Do as in Phase 1, lines 19–21 to interpolate $S_e^{(i)}, T_e^{(i)}$.

    for k from 0 to $L-1$ **do**

      If $i \stackrel{?}{=} 1$, set $P_e^{(i)}(k) = 1$; otherwise set $P_e^{(i)}(k) = 0$.

    for k from L to $2L$ **do**

      Sample share: $P_e^{(i)}(k) \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$.

      if $i \stackrel{?}{=} 1$ **then**

        Adjust first share: $P_e^{(i)}(k) \leftarrow P_e^{(i)}(k) + \Delta P_e(k)$.

    Interpolate $P_e^{(i)}$ and compute $c_e^{(i)} \leftarrow P_e^{(i)}(R_e)$.

    Compute $a_e^{(i)} \leftarrow S_e^{(i)}(R_e)$ and $b_e^{(i)} \leftarrow T_e^{(i)}(R_e)$.

  Compute $P_e(R_e) \leftarrow S_e(R_e) \cdot T_e(R_e)$.

  Compute missing output broadcast $\mathsf{ct}_e^{(\bar{i}_e)} = \mathsf{ct} - \sum_{i\neq\bar{i}_e} \mathsf{ct}_e^{(i)}$.

  Compute missing shares $a_e^{(\bar{i}_e)} \leftarrow S_e(R_e) - \sum_{i\neq\bar{i}_e} a_e^{(i)}$, $b_e^{(\bar{i}_e)} \leftarrow T_e(R_e) - \sum_{i\neq\bar{i}_e} b_e^{(i)}$ and $c_e^{(\bar{i}_e)} \leftarrow P_e(R_e) - \sum_{i\neq\bar{i}_e} c_e^{(i)}$.

Set $h_1' \leftarrow H_1\left(\begin{matrix}\mathsf{msg}, \mathsf{pk}, \mathsf{salt}, ((\mathsf{com}_e^{(i)})_{i\in[N]}, (\mathsf{ct}_e^{(i)})_{i\in[N]},\\ \Delta\mathsf{sk}_e, (\Delta t_{e,\ell})_{\ell\in[L]}, (\Delta P_e(k))_{k\in[L,2L]})_{e\in[\tau]}\end{matrix}\right)$.

Set $h_2' \leftarrow H_2\left(\begin{matrix}\mathsf{salt}, h_1', (S_e(R_e), T_e(R_e), P_e(R_e),\\ (a_e^{(i)}, b_e^{(i)}, c_e^{(i)})_{i\in[N]})_{e\in[\tau]}\end{matrix}\right)$.

Output accept iff $h_1' \stackrel{?}{=} h_1$ and $h_2' \stackrel{?}{=} h_2$.

Figure 8: Rainier Verification algorithm.

Proof. See Appendix A for the full proof.

6 Performance Evaluation & Comparison with Other Designs

We now compare the performance in terms of signature sizes and signing/verification run-times of several post-quantum signature schemes. To ensure a fair comparison, all candidates are benchmarked on the same machine, a standard Intel desktop i7-4790 CPU @ 3.60GHz. For Banquet [BdSGK+21], we used the public implementation [Ban21], which we also used as a starting point for our own C++ implementations of Banquet using the single-key Even–Mansour variants, LSAES and Rainier[5]. For other signature schemes, we use their implementation from the SUPERCOP[6] benchmarking suite, version `20210125`, if available.

6.1 Performance Evaluation

We compare the designs explored in this work to several other post-quantum signature schemes, namely the ones in the third round of the NIST PQC standardization project; of these, the ones with the most similarity are Picnic [ZCD+20], which is also an MPCitH-based scheme, and SPHINCS+[HBD+20] which also only relyies on the security of symmetric-key primitives.

We focus our comparisons on the 128-bit security level (NIST level L1), as this is expected to be sufficiently secure in the near term, and due to the fact that some of the constructions like EM-AES are not able to instantiate higher security levels in a straightforward way. We give some additional data for the 192-bit and 256-bit security levels in Appendix D.1.

In Figure 9, we give a graphical overview of the signing performance of the schemes explored in this work and compare them to two other schemes based on symmetric-key primitives, Picnic and SPHINCS+. For almost all of the schemes in Figure 9, verification has similar performance to signing, except for SPHINCS+, where the verification is much faster, in the range of 0.2-2.6ms, depending on the instance and used hash function. We give the numerical data and detailed parameter sets in Appendix D, Table 3.

Figure 9 shows that Rainier$_3$ and the more conservative Rainier$_4$ (based on Rain$_3$ and Rain$_4$, respectively) outperform the other schemes considerably. The flexible parametrization of MPCitH-based signatures results in a large array of possible instances that have varying tradeoffs between signature size and performance. When compared to the small SPHINCS+ variants, Rainier can offer signatures with similar size but two orders of magnitude faster signing or can offer similar signing performance while reducing signature sizes from 8 to 5 KB. Comparing to Picnic3, we can also have signatures that are less than half the size with the same performance. In Figure 9 we also see the expected improvements from our investigation of single-key Even-Mansour (EM) and LSAES. We see approximately 10% improvement in signature sizes (and running times) when using the EM variants, since the evaluation of the AES key schedule in the MPCitH protocol can be done publicly. The instances using LSAES-128 also show a similar improvement compared to using AES-128, and the EM-LSAES instances can have signatures below 10 KB.

[5]Our implementations are available at `https://github.com/IAIK/rainier-signatures`.
[6]https://bench.cr.yp.to/supercop.html

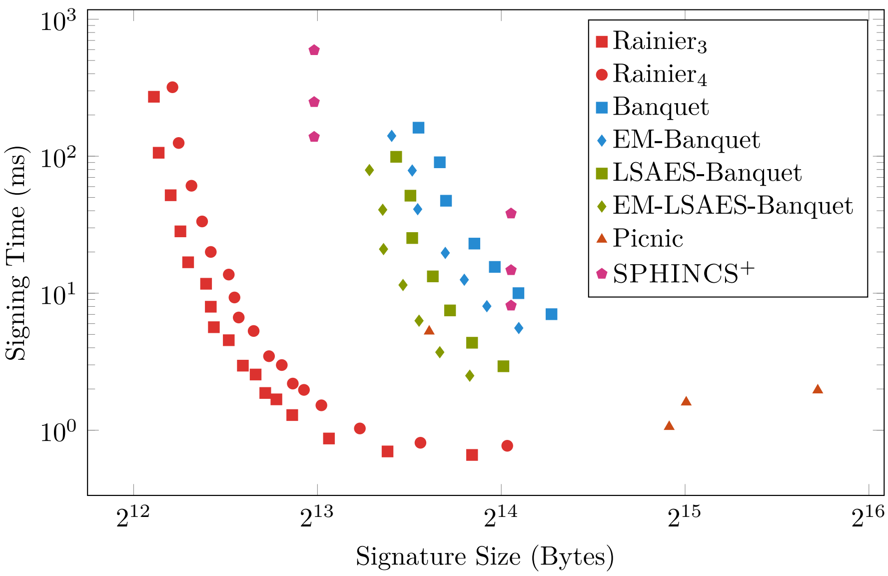


Figure 9: Comparison of signing time and signature size of various schemes at the 128-bit security level. Picnic instances include all proposed third round parameter sets for the L1 security level. SPHINCS$^+$ instances include all `simple` parameter sets for the L1 security level (`haraka`,`sha256`,`shake256`, in order of decreasing performance).

We compare Banquet and Rainier$_3$ to the third-round candidates in the NIST PQC project in Table 2. We also highlight the public key sizes since in a PKI environment, an X509 certificate includes both a signature and a public key. In this scenario the combined size of public key + signature is relevant, benefitting the signature schemes based on symmetric-key primitives, that have very small public keys, allowing them to somewhat offset their larger signatures.

As Table 2 shows, the lattice based schemes Dilithium and Falcon provide the best combined sizes for public key and signature, while also having very fast signing and verification operations. The multivariate schemes Rainbow and GeMSS suffer in this scenario due to their large public keys, however Rainbow has the fastest signing and verification times. Rainier provides smaller certificate sizes than Picnic and SPHINCS$^+$, and can also improve upon their signing times, which SPHINCS$^+$ providing faster verification times for their small parameter set than similar sized Rainier instances.

Other One-Way Function Designs. We discuss using alternative one-way functions such as *Vision* in MPCitH-based signatures in Appendix D.2.

7 Conclusions and Future Work

In this work, we present MPCitH-based signature schemes that produce – to the best of our knowledge – the smallest signature sizes when compared to existing schemes in this category. In addition, we open up or reinforce several exciting

Scheme	$\lvert\mathsf{pk}\rvert$	**Sig. size**	**Sign**	**Verify**
Picnic1-L1-FS [ZCD+20]	32	32 860	1.60	1.31
Picnic3-L1 [ZCD+20]	32	12 468	5.27	3.99
sphincss128sha256simple [HBD+20]	32	8 080	248.37	0.75
sphincsf128sha256simple [HBD+20]	32	16 976	14.73	1.79
Dilithium2 [LDK+20]	1 312	2 420	0.07	0.03
Falcon-512 [PFH+20]	897	666	0.11	0.02
Rainbow Ia-Classic [DCP+20]	161 600	66	0.02	0.01
GeMSS128v2 [CFM+20]	352 188	33	320.99	0.08
Banquet-AES-128 [BdSGK+21]	32	19 776	7.03	5.32
Banquet-AES-128 [BdSGK+21]	32	13 284	47.31	43.03
Rainier$_3$-128 ($N = 16, \tau = 33$)	32	8 544	0.87	0.81
Rainier$_3$-128 ($N = 107, \tau = 20$)	32	6 176	2.96	2.92
Rainier$_3$-128 ($N = 1624, \tau = 14$)	32	4 880	28.28	28.16
Rainier$_3$-128 ($N = 7121, \tau = 11$)	32	4 496	105.98	105.15

Table 2: Comparison of public-key and signature sizes at the 128-bit security level for the third-round candidates of the NIST PQC standardization project and the designs explored in this work. Size in bytes, time in ms.

directions for future research.

When implementing Rainier, an implementation of the multiplication in $\mathrm{GF}(2^n)$ for relatively large n is needed. The same is true for signature schemes based on elliptic curves over binary fields. Hence, basing Rainier on the same field could utilize shared resources, e.g., use the same multiplier in hardware. Therefore, it seems worthwhile to explore the overhead that implementations of a hybrid signature scheme of Rainier and a signature scheme using elliptic curves over binary fields bear compared to implementations that just enable either of them. Hence, we would instantiate Rain for the field $\mathrm{GF}(2^{233})$ defined by $X^{233}+X^{74}+1$ for 192 bits of security or use $\mathrm{GF}(2^{283})$ defined by $X^{283} + X^{12} + X^7 + X^5 + 1$ as given by NIST [KG13] for 256 bits of security.

Since we profit from using AES in Even–Mansour mode, it is of interest to evaluate which distinguisher on round-reduced AES can actually speed up key recovery attacks on round-reduced AES in Even–Mansour mode only having a single (plaintext, ciphertext) pair. In this OWF security setting, is it easier to recover the key of round-reduced AES or round-reduced AES in Even–Mansour mode? What is the difference in the number of rounds that can be attacked? We also note that when AES is used in EM mode for signature key pair generation, since the key used to define the permutation is public, the key schedule used to derive round keys may be far more complex (e.g., it could be SHAKE), without increasing signature size since it is no longer evaluated in the MPC computation. Does this improve the security of the EM construction?

Since MPCitH-based signature schemes are built from interactive proof protocols, it is possible to use them interactively in certain authentication scenarios (e.g., smart cards). This has been investigated for ZKBoo, Picnic, and Banquet in [GMO16, KZ20b, BdSGK+21]. Performance in this scenario is roughly an order of magnitude better than the non-interactive case, is the same true of Rainier?

The number of rounds we use for LSAES in Table 3 is the same as AES (e.g., 10 at the 128-bit security level). For OWF security, this seems to provide ample margin, and reduced-round variants would offer better performance. So this raises the question of how many rounds are sufficient for OWF security of LSAES and EM-LSAES with 32-bit and larger S-boxes?

Another interesting topic is the number of rounds of Rain. Indeed, it is fixed for all block sizes since we expect the strength of the inversion operation and the linear layer to scale with the block size and provide sufficient security when increasing the security level. Similarly, we expect that Rain does not get weaker when reducing the block size to toy levels. However, it is still interesting to see how Rain behaves when its components are "weakened". For instance, it would be interesting to conduct more research on the effects on security when removing round constants or using matrices with more structure.

Finally, we only consider attacks that use a single data pair. Regarding Rain, a possible direction for future work would be to determine the number of rounds that can be attacked when we increase the data complexity to the full codebook.

References

[AAB+20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3):1–45, 2020.

[ABC+20] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[ACG+19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. Cryptology ePrint Archive, Report 2019/419, 2019. to appear at ASIACRYPT 2019.

[AD18] Tomer Ashur and Siemen Dhooghe. MARVELlous: a STARK-Friendly Family of Cryptographic Primitives. *IACR Cryptol. ePrint Arch.*, 2018:1098, 2018.

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for MPC, and more. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part II*, volume 11736 of *LNCS*, pages 151–171. Springer, Heidelberg, September 2019.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 191–219, 2016.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.

[Ban21] Banquet: Short and Fast Signatures from AES, 2021. Software implementation from [BdSGK+21]. `https://github.com/dkales/banquet`.

[BBdS+20] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Alzette: A 64-bit ARX-box - (feat. CRAX and TRAX). In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 419–448. Springer, Heidelberg, August 2020.

[Bd20] Ward Beullens and Cyprien de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In *Post-Quantum Cryptography*, pages 130–150. Springer, 2020.

[BDF11] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 169–187. Springer, Heidelberg, August 2011.

[BdSGK+21] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In *PKC*, 2021. To appear.

[BHK+19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2129–2146. ACM Press, November 2019.

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[BHN+19] Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum attacks without superposition queries: The offline Simon's algorithm. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 552–583. Springer, Heidelberg, December 2019.

[BR19a] Navid Ghaedi Bardeh and Sondre Rønjom. The exchange attack: How to distinguish six rounds of AES with $2^{88.2}$ chosen plaintexts. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 347–370. Springer, Heidelberg, December 2019.

[BR19b] Navid Ghaedi Bardeh and Sondre Rønjom. Practical attacks on reduced-round AES. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 297–310. Springer, Heidelberg, July 2019.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 2–21. Springer, Heidelberg, August 1991.

[BS20] Dan Boneh and Victor Shoup. A graduate course in applied cryptography, 2020. Available online `https://crypto.stanford.edu/~dabo/cryptobook/`.

[Car63] L. Carlitz. A Note on the Betti-Mathieu group. *Portugaliae Mathematica*, 22:121–125, 1963.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

[CDG+20] Melissa Chase, David Derler, Steven Goldfeder, Jonathan Katz, Valdimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Xiao Wang, and Greg Zaverucha. The Picnic Signature Scheme Design Document (version 2.2), 2020.

[CDGP93] Luc J. M. Claesen, Joan Daemen, Mark Genoe, and G. Peeters. Subterranean: A 600 mbit/sec cryptographic VLSI chip. In *ICCD*, pages 610–613. IEEE Computer Society, 1993.

[CFM+20] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. GeMSS. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[CL05] Carlos Cid and Gaëtan Leurent. An analysis of the XSL algorithm. In Bimal K. Roy, editor, *ASIACRYPT*, volume 3788 of *LNCS*, pages 333–352. Springer, 2005.

[CMR06] Carlos Cid, Sean Murphy, and Matthew J. B. Robshaw. *Algebraic aspects of the advanced encryption standard.* Springer, 2006.

[CP02] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *LNCS*, pages 267–287. Springer, 2002.

[Dae95] Joan Daemen. *Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis.* K.U.Leuven, 1995.

[DCP+20] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. Rainbow. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[dDOS19] Cyprien de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. BBQ: Using AES in Picnic signatures. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 669–692. Springer, Heidelberg, August 2019.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low ANDdepth and few ANDs per bit. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 662–692. Springer, Heidelberg, August 2018.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 157–184. Springer, Heidelberg, August 2016.

[DFJ13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 371–387. Springer, Heidelberg, May 2013.

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round Fiat-Shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 602–631. Springer, Heidelberg, August 2020.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Heidelberg, August 2019.

[Dic01] Leonard Eugene Dickson. *Linear groups: with an exposition of the Galois field theory*, volume 6. BG Teubner, 1901.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The Block Cipher Square. In *FSE 1997*, volume 1267 of *LNCS*, pages 149–165, 1997.

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 336–354. Springer, Heidelberg, April 2012.

[DMMR20] Joan Daemen, Pedro Maat Costa Massolino, Alireza Mehrdad, and Yann Rotella. The Subterranean 2.0 cipher suite. *IACR Trans. Symm. Cryptol.*, 2020(S1):262–294, 2020.

[DMRS20] Christoph Dobraunig, Farokhlagha Moazami, Christian Rechberger, and Hadi Soleimany. Framework for faster key search using related-key higher-order differential properties: Applications to Agrasta. *IET Inf. Secur.*, 14(2):202–209, 2020.

[DR98] Joan Daemen and Vincent Rijmen. The block cipher Rijndael. In *CARDIS*, volume 1820 of *LNCS*, pages 277–284. Springer, 1998.

[DR20] Joan Daemen and Vincent Rijmen. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition.* Information Security and Cryptography. Springer, 2020.

[DS08] Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 116–126. Springer, Heidelberg, February 2008.

[dSGOT21] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge mpcith-based arguments. *IACR Cryptol. ePrint Arch.*, 2021:215, 2021.

[EGL+20] Maria Eichlseder, Lorenzo Grassi, Reinhard Lüftenegger, Morten Øygarden, Christian Rechberger, Markus Schofnegger, and Qingju Wang. An algebraic attack on ciphers with low-degree round functions: Application to full MiMC. In *ASIACRYPT (1)*, volume 12491 of *LNCS*, pages 477–506. Springer, 2020.

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–162, June 1997.

[FGLM93] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.

[FJM14] Pierre-Alain Fouque, Antoine Joux, and Chrysanthi Mavromati. Multi-user collisions: Applications to discrete logarithm, Even-Mansour and PRINCE. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 420–438. Springer, Heidelberg, December 2014.

[FKL+01] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In Bruce Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 213–230. Springer, Heidelberg, April 2001.

[GCZ16] Steven Goldfeder, Melissa Chase, and Greg Zaverucha. Efficient post-quantum zero-knowledge and signatures. Cryptology ePrint Archive, Report 2016/1110, 2016. `http://eprint.iacr.org/2016/1110`.

[Gen07] Giulio Genovese. Improving the algorithms of berlekamp and niederreiter for factoring polynomials over finite fields. *J. Symb. Comput.*, 42(1-2):159–177, 2007.

[GHHM20] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. Cryptology ePrint Archive, Report 2020/1361, 2020. `https://eprint.iacr.org/2020/1361`.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX Security Symposium*, Vancouver, B.C., August 2021. USENIX Association.

[GLR+20] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a generalization of substitution-permutation networks: The HADES design strategy. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 674–704. Springer, Heidelberg, May 2020.

[GM16] Shay Gueron and Nicky Mouha. Simpira v2: A family of efficient permutations using the AES round function. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 95–125. Springer, Heidelberg, December 2016.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016.

[Gol07] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2007.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM STOC*, pages 212–219. ACM Press, May 1996.

[GRR17] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. A new structural-differential property of 5-round AES. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 289–317. Springer, Heidelberg, April / May 2017.

[HBD+20] Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[HS18] Akinori Hosoyamada and Yu Sasaki. Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 198–218. Springer, Heidelberg, April 2018.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

[JAC+20] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing Grover oracles for quantum key search on AES and LowMC. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 280–310. Springer, Heidelberg, May 2020.

[JV17] Antoine Joux and Vanessa Vitse. A crossbred algorithm for solving boolean polynomial systems. In *NuTMiC*, volume 10737 of *LNCS*, pages 3–21, 2017.

[Kat10] Jonathan Katz. *Digital signatures.* Springer Science & Business Media, 2010.

[KG13] Cameron F. Kerry and Patrick D. Gallagher. FIPS PUB 186-4: Digital Signature Standard (DSS), 2013.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

[KLMR16] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - Efficient short-input hashing for post-quantum applications. *IACR Trans. Symm. Cryptol.*, 2016(2):1–29, 2016. `http://tosc.iacr.org/index.php/ToSC/article/view/563`.

[KLSW17] Thorsten Kranz, Gregor Leander, Ko Stoffelen, and Friedrich Wiemer. Shorter linear straight-line programs for MDS matrices. *IACR Trans. Symm. Cryptol.*, 2017(4):188–211, 2017.

[KZ20a] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 3–22. Springer, Heidelberg, December 2020.

[KZ20b] Daniel Kales and Greg Zaverucha. Improving the performance of the Picnic signature scheme. *IACR TCHES*, 2020(4):154–188, 2020. `https://tches.iacr.org/index.php/TCHES/article/view/8680`.

[Lai94] Xuejia Lai. *Higher Order Derivatives and Differential Cryptanalysis*, pages 227–233. 1994.

[LDK+20] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[LN96] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 1996.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 386–397. Springer, Heidelberg, May 1994.

[MDRMH10] Hamid Mala, Mohammad Dakhilalian, Vincent Rijmen, and Mahmoud Modarres-Hashemi. Improved impossible differential cryptanalysis of 7-round AES-128. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 282–291. Springer, Heidelberg, December 2010.

[ML15] Nicky Mouha and Atul Luykx. Multi-key security: The Even-Mansour construction revisited. In Rosario Gennaro and Matthew

J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 209–223. Springer, Heidelberg, August 2015.

[MR02] Sean Murphy and Matthew J. B. Robshaw. Essential algebraic structure within the AES. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 1–16. Springer, 2002.

[MV04] David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Heidelberg, December 2004.

[NIS15] NIST. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. National Institute of Standards and Technology (NIST), FIPS PUB 202, U.S. Department of Commerce, 2015.

[NK93] Kaisa Nyberg and Lars R. Knudsen. Provable security against differential cryptanalysis (rump session). In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 566–574. Springer, Heidelberg, August 1993.

[NNY18] Ruben Niederhagen, Kai-Chun Ning, and Bo-Yin Yang. Implementing Joux-Vitse's crossbred algorithm for solving $\mathcal{MQ}$ systems over $\mathbb{F}_2$ on GPUs. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 121–141. Springer, Heidelberg, 2018.

[Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 55–64. Springer, Heidelberg, May 1994.

[PFH$^+$20] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[RDP$^+$96] Vincent Rijmen, Joan Daemen, Bart Preneel, Anton Bossalaers, and Erik De Win. The cipher SHARK. In Dieter Gollmann, editor, *FSE'96*, volume 1039 of *LNCS*, pages 99–111. Springer, Heidelberg, February 1996.

[Riv84] Ron Rivest. DESX, 1984. Unpublished.

[WL13] Baofeng Wu and Zhuojun Liu. Linearized polynomials over finite fields revisited. *Finite Fields Their Appl.*, 22:79–100, 2013.

[ZCD$^+$20] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao

Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

A Security Proofs

Since the Rainier signature scheme is similar to Banquet the security analysis is also similar. The schemes are just different enough that the results do not directly apply, necessitating a separate analysis for Rainier. That said, this section is a simplified version of the corresponding section in [BdSGK+21] where the scheme and analysis are modified for a 5-round protocol, rather than the 7-round one as in [BdSGK+21]. Many parts of the analysis are identical, and we give full credit to [BdSGK+21].

In Theorem 4, we prove that Rainier is an $\mathsf{EUF\text{-}CMA}$ secure signature scheme. Our definition of unforgeability under chosen message attacks is the standard one, as defined in [Kat10, Definition 1.6]. We first prove in Lemma 5 that Rainier is $\mathsf{EUF\text{-}KO}$ secure, i.e., secure against forgery attacks where the attacker is only given the public key, and no signature queries. A formal definition is obtained from the $\mathsf{EUF\text{-}CMA}$ defintion by removing the adversary's access to the signing oracle. The idea is that because the protocol is sound, if an attacker successfully creates a forgery, then by reading the random oracle query history, we can extract the secret key, inverting the one-way function used in key generation.

Then to show that the scheme is $\mathsf{EUF\text{-}CMA}$ secure in Theorem 4, we additionally show that signatures may be simulated without knowledge of the private key, by programming the random oracles.

All adversaries in this section are assumed to be probabilistic polynomial time (in κ) algorithms.

Lemma 5. *Let Commit, H_1 and H_2 be modeled as random oracles, $\mathsf{Expand}()$ be modeled as a random function, and let (N, τ, κ, L) be parameters of the Rainier signature scheme. Let $\mathcal{A}$ be an adversary against the $\mathsf{EUF\text{-}KO}$ security of Rainier that makes a total of Q random oracle queries. Assuming that KeyGen is an ε_{OWF}-hard one-way function, then $\mathcal{A}$'s advantage in the $\mathsf{EUF\text{-}KO}$ game is*

$$\varepsilon_{KO} \leq \varepsilon_{OWF} + \frac{(\tau N + 1)Q^2}{2^{2\kappa}} + \Pr[X + Y = \tau],$$

where $\Pr[X + Y = \tau]$ is as described in the proof.

Remark 6. We do not express $\Pr[X + Y = \tau]$ as a closed function; we must choose parameters (N, τ, L) for Rainier such that it is negligible in κ.

Proof. We give an algorithm $\mathcal{B}$ which uses the EUF-KO adversary $\mathcal{A}$ to compute a pre-image for the key generation OWF.

Algorithm $\mathcal{B}$ simulates the EUF-KO game using the random oracles H_{c} (shorthand for Commit), H_1 and H_2 and query lists $\mathcal{Q}_{\mathsf{c}}, \mathcal{Q}_1$ and $\mathcal{Q}_2$. In addition, $\mathcal{B}$ maintains three tables $\mathcal{T}_{\mathsf{sh}}, \mathcal{T}_{\mathsf{in}}$ and $\mathcal{T}_{\mathsf{op}}$ to store shares of the parties, inputs to the MPC protocol and openings of the polynomial checking protocol that it recovers from $\mathcal{A}$'s RO queries. $\mathcal{B}$ also maintains a set Bad to keep track of the outputs of all three random oracles. We also ignore calls to $\mathsf{Expand}()$ in our

Algorithm 1 $H_{\mathsf{c}}(q_{\mathsf{c}} = (\mathsf{salt}, e, i, \mathsf{seed}))$:

1: $x \xleftarrow{\$} \{0,1\}^{2\kappa}$.
2: **if** $x \in \mathsf{Bad}$ **then** abort. ▷ Check if x is fresh.
3: $x \to \mathsf{Bad}$.
4: $(q_{\mathsf{c}}, x) \to \mathcal{Q}_{\mathsf{c}}$.
5: Return x.

analysis, since they are used to expand outputs from H_1 and H_2 when $\mathsf{Expand}()$ is a random function this is equivalent to increasing the output lengths of H_1 and H_2.

Behavior of $\mathcal{B}$. On input pk, a OWF challenge, algorithm $\mathcal{B}$ forwards it to $\mathcal{A}$ as a Rainier public key for the $\mathsf{EUF\text{-}KO}$ game. It lets $\mathcal{A}$ run and answers its random oracle queries in the following way. We assume (wlog.) that Algorithm 1, Algorithm 2 and Algorithm 3 only consider queries that are correctly formed, and ignore duplicate queries.)

- H_{c}: When $\mathcal{A}$ queries the commitment random oracle, $\mathcal{B}$ records the query to learn which commitment corresponds to which seed. See Algorithm 1.
- H_1: When $\mathcal{A}$ commits to seeds and sends the offsets for the secret key and the inverse values, $\mathcal{B}$ checks whether the commitments were output by its simulation of H_{c}. If any were for some e and i, then $\mathcal{B}$ is able to reconstruct the shares for party i in execution e. If $\mathcal{B}$ was able to reconstruct every party's share for any e, then it can use the offsets included in σ_1 to extract the values used by $\mathcal{A}$ in that execution. For the checking polynomials, $\mathcal{B}$ uses the newly sampled response to expand the challenges and extract the checking polynomials. See Algorithm 2.
- H_2: No extraction takes place during this random oracle simulation. See Algorithm 3.

When $\mathcal{A}$ terminates, $\mathcal{B}$ checks the $\mathcal{T}_{\mathsf{in}}$ table for any entry where the extracted sk_e is consistent with pk. If a match is found, $\mathcal{B}$ outputs sk_e as a pre-image for the OWF, otherwise $\mathcal{B}$ outputs $\perp$.

Advantage of the reduction. Given the behavior presented above, we have the following by the law of total probability:

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] =& \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ aborts}] + \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ outputs } \perp] \\ &+ \Pr[\mathcal{A} \text{ wins} \wedge \mathcal{B} \text{ outputs } \mathsf{sk}] \\ \leq& \Pr[\mathcal{B} \text{ aborts}] + \Pr[\mathcal{A} \text{ wins} \mid \mathcal{B} \text{ outputs } \perp] \\ &+ \Pr[\mathcal{B} \text{ outputs } \mathsf{sk}]. \end{aligned} \tag{7}$$

Let Q_{com}, Q_1 and Q_2 denote the number of queries made by $\mathcal{A}$ to each respective random oracle. Given the way in which values are added to Bad, we

Algorithm 2 $H_1(q_1 = \sigma_1)$:

Parse σ_1 as $(\mathsf{salt}, ((\mathsf{com}_e^{(i)})_{i\in[N]}, (\mathsf{ct}_e^{(i)})_{i\in[N]}, \Delta\mathsf{sk}_e, (\Delta t_{e,\ell})_{\ell\in[L]}, (\Delta P_e(k))_{k\in[L,2L]})_{e\in[\tau]})$

for $e \in [\tau], i \in [N]$ **do** $\mathsf{com}_e^{(i)} \to \mathsf{Bad}$.

If the committed seed is known for a certain e, i, then $\mathcal{B}$ records the shares of the secret key and of the inverse values from that party that derive from that seed and the offsets committed to in σ_1:

for $(e, i) \in [\tau] \times [N] : \exists\, \mathsf{seed}_e^{(i)} : ((\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)}), \mathsf{com}_e^{(i)}) \in \mathcal{Q}_\mathsf{c}$ **do**

  $\mathsf{sk}_e^{(i)}, (t_{e,\ell}^{(i)})_\ell \leftarrow \mathsf{ExpandTape}(\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)})$.

  if $i \stackrel{?}{=} 1$ **then** $\mathsf{sk}_e^{(i)} \leftarrow \mathsf{sk}_e^{(i)} + \Delta\mathsf{sk}_e$ and $(t_{e,\ell}^{(i)})_\ell \leftarrow (t_{e,\ell}^{(i)} + \Delta t_{e,\ell})_\ell$.

  $(\mathsf{sk}_e^{(i)}, (t_{e,\ell}^{(i)})_\ell) \to \mathcal{T}_\mathsf{sh}[q_1, e, i]$.

If the shares of the secret key and of the inverse values are known for every party in that execution, $\mathcal{B}$ records the resulting secret key and inverse values:

for each $e : \forall i, \mathcal{T}_\mathsf{sh}[q_1, e, i] \neq \emptyset$ **do**

  $\mathsf{sk}_e \leftarrow \sum_i \mathsf{sk}_e^{(i)}$ and $(t_{e,\ell})_\ell \leftarrow (\sum_i t_{e,\ell}^{(i)})_\ell$.

  $(\mathsf{sk}_e, (t_{e,\ell})_\ell) \to \mathcal{T}_\mathsf{in}[q_1, e]$.

$x \xleftarrow{\$} \{0,1\}^{2\kappa}$.

if $x \in \mathsf{Bad}$ **then** abort.

$x \to \mathsf{Bad}$.

$(q_1, x) \to \mathcal{Q}_1$.

Store the multiplication checking values.

$(R_e)_e \leftarrow \mathsf{Expand}(x)$.

for each $e : \mathcal{T}_\mathsf{in}[q_1^*, e] \neq \emptyset$ **do**

  $(P_e(R_e), S_e(R_e), T_e(R_e))_e \to \mathcal{T}_\mathsf{op}[q_2, e]$.

Return x.

have:

$$\begin{aligned}
\Pr[\mathcal{B} \text{ aborts}] &= (\#\text{times an } x \text{ is sampled}) \cdot \Pr[\mathcal{B} \text{ aborts at that sample}] \\
&\leq (Q_\mathsf{com} + Q_1 + Q_2) \cdot \frac{\max|\mathsf{Bad}|}{2^{2\kappa}} \\
&= (Q_\mathsf{com} + Q_1 + Q_2) \cdot \frac{Q_\mathsf{com} + (\tau N + 1)Q_1 + 2Q_2}{2^{2\kappa}} \\
&\leq \frac{(\tau N + 1)(Q_\mathsf{com} + Q_1 + Q_2)^2}{2^{2\kappa}}.
\end{aligned} \tag{8}$$

We now analyze the probability of $\mathcal{A}$ winning the EUF-KO experiment conditioned on the event that $\mathcal{B}$ outputs $\perp$, i.e., no pre-image to pk was found on the query lists.

Cheating in the first round. For any query $q_1 \in \mathcal{Q}_1$, and its corresponding answer $h_1 = (R_e)_{e\in[\tau]}$, let $G_1(q_1, h_1)$ be the set of indices $e \in [\tau]$ of "good executions" where both $\mathcal{T}_\mathsf{in}[q_1, e] = (\mathsf{sk}_e, (t_{e,\ell})_{\ell\in[L]})$ is non-empty and and it holds that

$$P_e(R_e) = S_e(R_e) \cdot T_e(R_e). \tag{9}$$

If there does not exist such a q_1, let $G_1(q_1, h_1) = \emptyset$.

Algorithm 3 $H_2(q_2 = (h_1, \sigma_2))$:

1: $h_1 \to \mathsf{Bad}$.
2: $x \xleftarrow{\$} \{0,1\}^{2\kappa}$.
3: **if** $x \in \mathsf{Bad}$ **then** abort.
4: $x \to \mathsf{Bad}$.
5: $(q_2, x) \to \mathcal{Q}_2$.
6: Return x.

For any such good execution $e \in G_2(q_2, h_2)$, since $\mathcal{B}$ outputs $\perp$ but $\mathcal{A}$ wins, this implies that either the challenges in the first round were such that Eq. (9) held (in which case any value of R_e passes the check), or the challenge R_e was sampled such that Eq. (9) held. Conditioning on the first event not happening, Lemma 3 gives us that the second happens with probability at most $p_1 := 2L/(2^\kappa - 2L)$, given that h_1 is distributed uniformly at random (which holds assuming H_1 and $\mathsf{Expand}()$ are random functions).

As the response h_1 is uniform, each $e \in [\tau]$ has the same independent probability of being in $G_1(q_1, h_1)$, given that $\mathcal{B}$ outputs $\perp$. We therefore have that $\#G_1(q_1, h_1) \mid_\perp \sim X_{q_1}$ where $X_{q_1} = \mathfrak{B}(\tau, p_1)$, where $\mathfrak{B}(\tau, p_1)$ is the binomial distribution with τ events, each with success probability p_1. Letting $(q_{\mathsf{best}_1}, h_{\mathsf{best}_1})$ denote the query-response pair which maximizes $\#G_1(q_1, h_1)$, we then have that

$$\#G_1(q_{\mathsf{best}_1}, h_{\mathsf{best}_1}) \mid_\perp \sim X = \max_{q_1 \in \mathcal{Q}_1} \{X_{q_1}\}.$$

Cheating in the second round. Each second round query $q_2 = (h_1, \sigma_2)$ that $\mathcal{A}$ makes to H_2 can only be used in a valid signature if there exists a corresponding query $(q_1, h_1) \in \mathcal{Q}_1$. Then for each "bad" first-round execution $e \in [\tau] \backslash G_1(q_1, h_1)$, either verification failed, in which case $\mathcal{A}$ couldn't have won, or the verification passed, despite Eq. (9) not being satisfied. This implies that exactly one of the parties must have cheated. At least one cheater is required for verification to pass, but as $N - 1$ parties are opened, verification would fail if more than one party cheated.

Since $h_2 \in [N]^\tau$ is distributed uniformly at random, the probability that this happens for all such "bad" first-round executions e is

$$\left(\frac{1}{N}\right)^{\tau - \#G_1(q_1, h_1)} \leq \left(\frac{1}{N}\right)^{\tau - \#G_1(q_{\mathsf{best}_1}, h_{\mathsf{best}_1})}.$$

The probability that this happens for at least one of the Q_2 queries made to H_2 is

$$\Pr[\mathcal{A} \text{ wins} \mid \#G_1(q_{\mathsf{best}_1}, h_{\mathsf{best}_1}) = \tau_1] \leq 1 - \left(1 - \left(\frac{1}{N}\right)^{\tau - \tau_1}\right)^{Q_2}.$$

Finally conditioning on $\mathcal{B}$ outputting $\perp$ and summing over all values of τ_1, we have that

$$\Pr[\mathcal{A} \text{ wins} \mid \perp] \leq \Pr[X + Y = \tau] \tag{10}$$

where X is as before, and $Y = \max_{q_2 \in \mathcal{Q}_2}\{Y_{q_2}\}$ where the Y_{q_2} variables are independently and identically distributed as $\mathfrak{B}(\tau - X, 1/N))$.

Conclusion. Bringing Eq. (7), Eq. (8) and Eq. (10) together, we obtain the following.

$$\Pr[\mathcal{A}\text{ wins}] \leq \frac{(\tau N+1)(Q_{\mathsf{com}}+Q_1+Q_2)^2}{2^{2\kappa}} + \Pr[X+Y=\tau] + \Pr[\mathcal{B}\text{ outputs }\mathsf{sk}]$$

Assuming KeyGen is an ε_{OWF}-secure OWF and setting $Q = Q_{\mathsf{com}} + Q_1 + Q_2$ gives the required bound and concludes the proof. □

We assume that $\mathsf{ExpandTape}()$ is a secure pseudorandom generator (PRG), again using the standard definition, see for example [BS20, Definition 3.1]. In our implementation $\mathsf{ExpandTape}$ is implemented with the SHA-3 based extendable output function SHAKE [NIS15]. The assumption related to the tree derivation construction for random seeds is that it must be hiding. Informally, this means that after revealing a subset of the seeds (e.g., $N-1$ of N seeds), the remaining seeds remain hidden to a computationally bounded adversary. In [CDG+20, Section 6.3] it is shown that this holds when the hash function used to derived seeds is modelled as a random oracle. Secure one-way functions are defined in [Gol07, Section 2.2].

Theorem 4. The Rainier signature scheme is $\mathsf{EUF\text{-}CMA}$-secure, assuming that Commit, H_1, H_2 and Expand are modelled as random oracles, $\mathsf{ExpandTape}$ is a secure PRG, the seed tree construction is computationally hiding, the (N, τ, L) parameters are appropriately chosen, and that KeyGen is a secure one-way function.

Proof. Fix an attacker $\mathcal{A}$. We define a sequence of games where the first corresponds to $\mathcal{A}$ interacting with the real signature scheme in the $\mathsf{EUF\text{-}CMA}$ game. Through a series of hybrid arguments we show that this is indistinguishable from a simulated game, under the assumptions above. Let G_0 be the unmodified $\mathsf{EUF\text{-}CMA}$ game and let $\mathcal{B}$ denote an adversary against the $\mathsf{EUF\text{-}KO}$ game that acts as a simulator of the $\mathsf{EUF\text{-}CMA}$ game to $\mathcal{A}$. As we're in the random oracle model: when $\mathcal{A}$ queries one of its random oracles, $\mathcal{B}$ first checks if that query has been recorded before; if so, then it responds with the recorded answer; if not, $\mathcal{B}$ forwards the query to its corresponding random oracle, records the query and the answer it receives and forwards the answer to $\mathcal{A}$. Let G_i denote the probability that $\mathcal{A}$ succeeds in game G_i. At a high level, the sequence of games is as follows:

G_0: $\mathcal{B}$ knows a real secret key sk and can compute signatures honestly;

G_1: $\mathcal{B}$ replaces real signatures with simulated ones which no longer use sk;

$\mathcal{B}$ then uses the $\mathsf{EUF\text{-}KO}$ challenge pk^* in its simulation with $\mathcal{A}$.

We note that $\mathcal{A}$'s advantage in the $\mathsf{EUF\text{-}CMA}$ game is $\varepsilon_{\text{CMA}} = \mathsf{G}_0 = (\mathsf{G}_0 - \mathsf{G}_1) + \mathsf{G}_1$ and we obtain a bound on G_0 by first bounding $\mathsf{G}_0 - \mathsf{G}_1$ and then G_1

Hopping to Game G_1. When $\mathcal{A}$ queries the signing oracle, $\mathcal{B}$ simulates a signature by sampling a random secret key sk^*, choosing a party $\mathcal{P}_{i^*}$ at random and cheating in the verification phase and in the broadcast of the output shares $\mathsf{ct_e}^{(i)}$ such that the circuit still outputs the correct ciphertext, and finally ensuring that the values observed by $\mathcal{A}$ are sampled independently of sk^* and with a distribution that is computationally indistinguishable from a real signature. $\mathcal{B}$ programs H_1 to return the R_e values that it sampled, and H_2 to hide the cheating party $\mathcal{P}_{i^*}$ in Phase 5.

We now argue that the simulated signatures in G_1 are computationally indistinguishable from real signatures in G_0. We list a series of (sub) game hops which begins with G_0, where sk is known and signatures are created honestly, and ends with G_1, where signatures are simulated without using sk. With each change to $\mathcal{B}$'s behavior, we give an argument as to why the simulation remains indistinguishable, and quantify these below.

1. The initial $\mathcal{B}$ knows the real sk and can compute honest signatures as in the protocol. It only aborts if the salt that it samples in Phase 1 has already been queried. As its simulation is perfect, $\mathcal{B}$ is indistinguishable from the real $\mathsf{EUF\text{-}CMA}$ game as long as it does not abort.

2. Before beginning, the next $\mathcal{B}$ samples h_2 at random and expands it to obtain $(i^*_e)_{e\in[\tau]}$; these are the unopened parties, which $\mathcal{B}$ will use for cheating. It proceeds as before and programs the random oracle H_2 so that it outputs h_2 when queried in Phase 6. If that query has already been made, $\mathcal{B}$ aborts the simulation.

3. In Phase 1, the next $\mathcal{B}$ replaces $\mathsf{seed}_e^{(i^*)}$ in the binary tree, for each $e \in [\tau]$, by a randomly sampled one. This is indistinguishable from the previous version of $\mathcal{B}$ assuming that the tree structure is hiding.

4. The next $\mathcal{B}$ replaces the random tapes for party i^*, i.e., the outputs of $\mathsf{ExpandTape}(\mathsf{salt}, e, i^*, \mathsf{seed}_e^{(i^*)})$, by random outputs (independent of the seed). This is indistinguishable from the previous reduction assuming that $\mathsf{ExpandTape}()$ is a secure PRG.

5. The next $\mathcal{B}$ replaces the commitments of the unopened parties $\mathsf{com}_e^{(i^*)}$ with random values (i.e., without querying Commit). $\mathcal{B}$ aborts if $\mathcal{A}$ queries x such that $\mathsf{Commit}(x)$ was output by $\mathcal{B}$.

6. Before starting Phase 2, the next $\mathcal{B}$ samples h_1 at random and expands it to obtain $(R_e)_{e\in[\tau]}$; this will enable it to sample the checking values at random. It then proceeds as before and programs the random oracle H_1 to output h_1 in Phase 2. If that query has already been made, $\mathcal{B}$ aborts the simulation.

7. In Phase 1, the next $\mathcal{B}$ interpolates $S_e^{(i)}$ for $i \in [N] \setminus \{i^*\}$, samples the values $S_e(R_e)$ at random, computes $S_e^{(i^*)}(R_e) = S_e(R_e) - \sum_{i \neq i^*} S_e^{(i)}$ and interpolates $S_e^{(i^*)}$ using $k \in \{0, \dots, L-1\} \cup \{R_e\}$. It does the same for the T polynomials and computes P_e and the offsets according to the protocol. As the uniform distribution of honestly generated $S_e(R_e)$ and $T_e(R_e)$ (opened in Phase 3) comes from the uniform distribution of $\bar{s}_e$ and $\bar{t}_e$ read from the random tape (recall that $\mathsf{seed}_e^{(i^*)}$ is no longer used), this is indistinguishable from the previous hop. The same holds for the shares of party $\mathcal{P}_{i^*}$ that are opened in Phase 5. The distribution of the ΔP_e offsets is therefore also indistinguishable from a real signature as they are computed honestly

from indistinguishable elements. (At this stage the P_e polynomials always satisfy the check since $\mathcal{B}$ is still using a real sk.)

8. In Phase 5, the next $\mathcal{B}$ replaces $c_e^{(i^*)} \leftarrow P_e^{(i^*)}(R_e)$ with $c_e^{(i)} \leftarrow P_e(R_e) - \sum_{i \neq i^*} P_e^{(i)}(R_e)$. This is indistinguishable because the $P_e^{(i)}(R_e)$ values, for $i \neq i^*$, are computed honestly, and the $P_e(R_e)$ value is distributed identically to an honest signature (because $S_{e,j}$ and $T_{e,j}$ are). From now on, the Schwartz–Zippel check always passes, even if the product relation doesn't hold, and the distribution of everything that $\mathcal{A}$ can observe is indistinguishable from an honest signature and independent of hidden values.

9. The final $\mathcal{B}$ replaces the real sk by a random key sk^* and cheats on the broadcast of party P_{i^*}'s output share $\mathsf{ct}_e^{(i^*)}$ such that it matches what is expected, given the $N-1$ other shares. As $\mathsf{sk}_e^{(i^*)}$ is independent from the seeds $\mathcal{A}$ observes, the distribution of $\Delta\mathsf{sk}_e^*$ is identical and $\mathcal{A}$ has no information about sk^*. As $\mathcal{P}_{i^*}$ is never opened, $\mathcal{B}$'s cheating on $\mathsf{ct}_e^{(i^*)}$ can't be detected.

We can conclude that $\mathcal{B}$'s simulation of the signing oracle is indistinguishable and that $\mathcal{A}$ behaves exactly as in the real $\mathsf{EUF\text{-}CMA}$ game unless an abort happens.

There are four points at which $\mathcal{B}$ could abort: if the salt it sampled has been used before, if the committed value it replaces is queried, or if its queries to H_1 and H_2 have been made previously. Let Q_{salt} denote the number of different salts queried during the game (by both $\mathcal{A}$ and $\mathcal{B}$); each time $\mathcal{B}$ simulates a signature, it has a maximum probability of $Q_{\mathsf{salt}}/2^{2\kappa}$ of selecting an existing salt and aborting. Let Q_c denote the number of queries made to Commit by $\mathcal{A}$, including those made during signature queries. Since Commit is a random oracle, and $\mathsf{seed}_e^{(i^*)}$ is a uniformly random κ-bit value not used by $\mathcal{B}$ elsewhere, each time $\mathcal{B}$ attempts a new signature, it has a maximum probability of $Q_c/2^{\kappa}$ of replacing an existing commitment and aborting.

Similarly for H_1, resp. H_2, $\mathcal{B}$ has a maximum probability of $Q_1/2^{2\kappa}$, resp. $Q_2/2^{2\kappa}$ of aborting, where Q_1 and Q_2 denote the number of queries made to each random oracle during the game. Note that $\mathcal{B}$ samples one salt, replaces τ commitments and makes one query to both H_1 and H_2 for each signature query.

Let Q_s be the total number of signature queries, therefore

$$\mathsf{G}_0 - \mathsf{G}_1 \leq Q_s \cdot (\tau \cdot \varepsilon_{\text{PRG}} + \varepsilon_{\text{TREE}} + \Pr[\mathcal{B} \text{ aborts}])$$

where

$$\begin{aligned}
\Pr[\mathcal{B} \text{ aborts}] &\leq Q_{\mathsf{salt}}/2^{2\kappa} + Q_c/2^{\kappa} + Q_1/2^{2\kappa} + Q_2/2^{2\kappa} \\
&= (Q_{\mathsf{salt}} + Q_1 + Q_2)/2^{2\kappa} + Q_c/2^{\kappa} \\
&\leq (Q_1 + Q_2)/2^{2\kappa-1} + Q_c/2^{\kappa} \qquad (\text{Since } Q_{\mathsf{salt}} \leq Q_1 + Q_2)\ , \\
&\leq Q/2^{\kappa} \qquad\qquad (\text{where } Q = Q_1 + Q_2 + Q_c)\ .
\end{aligned}$$

Bounding G_1. In G_1, $\mathcal{B}$ is no longer using the witness and is instead simulating signatures only by programming the random oracles; it therefore replaces the honestly computed pk with and instance pk^* of the $\mathsf{EUF\text{-}KO}$ game. We see that if $\mathcal{A}$ wins G_1, i.e. outputs a valid signature, then $\mathcal{B}$ outputs a valid signature in the $\mathsf{EUF\text{-}KO}$ game, and so we have

$$\mathsf{G}_1 \leq \varepsilon_{\text{KO}} \leq \varepsilon_{\text{OWF}} + \frac{(\tau N + 1)Q^2}{2^{2\kappa}} + \Pr[X + Y = \tau],$$

where the bound on the advantage ε_{KO} of a $\mathsf{EUF\text{-}KO}$ attacker follows from Lemma 5. By a union bound, we have that

$$\begin{aligned}\varepsilon_{\text{CMA}} \leq \varepsilon_{\text{OWF}} + &\frac{(\tau N + 1)Q^2}{2^{2\kappa}} + \Pr[X + Y = \tau] \\ &+ Q_s \cdot (\tau \cdot \varepsilon_{\text{PRG}} + \varepsilon_{\text{TREE}} + Q/2^{\kappa}) .\end{aligned}$$

Assuming that $\mathsf{ExpandTape}()$ is a secure PRG that is ε_{PRG}-close to uniform, that the seed tree construction is hiding (so that $\varepsilon_{\text{TREE}}$ is negligible), that key generation is a one-way function and that parameters (N, τ, L) are appropriately chosen implies that ε_{CMA} is negligible in κ.

□

B Linear Layer and Key Schedule of LSAES

B.1 Linear layer L

The linear layer L consists the serial application of the linear function AB, SR, and MC, where as usual in descriptions of AES-128, the linear layer AB is seen as part of the description of the S-box. So we have $L(x) = MC \circ SR \circ AB(x)$.

The layer AB interprets all bytes x_i of the state as a series of bits $x_i = b_7 \| b_6 \| b_5 \| b_4 \| b_3 \| b_2 \| b_1 \| b_0$. On each of the resulting bit vectors, the following affine transformation is performed:

$$AB_i(x_i) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} .$$

The layer shift rows SR performs a row-wise rotation of the square state by the row index j to the left. So we get

$$SR(x) = x_0 \| x_1 \| x_2 \| x_3 \| x_5 \| x_9 \| x_{13} \| x_1 \| x_{10} \| x_{14} \| x_2 \| x_6 \| x_{15} \| x_3 \| x_7 \| x_{11} .$$

For the MixColumns layer, each byte of the state is interpreted as an element $\mathrm{GF}(2^8)$ using the irreducible polynomial $X^8 + X^4 + X^3 + X + 1$. On each of the rows h $(0 \leq h < 4)$, the following matrix multiplication is applied

$$MC_h(x_{0+h}, x_{1+h}, x_{2+h}, x_{3+h}) = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_{0+h} \\ x_{1+h} \\ x_{2+h} \\ x_{3+h} \end{pmatrix} .$$

B.2 Key Schedule

Given our secret key k we interpret it as as bytes and immediately get our first round key $k^{(1)} = k = k_0^{(1)} \| k_1^{(1)} \| \dots \| k_{15}^{(1)}$. To compute round key $k^{(i+1)}$ from $k^{(i)}$, we do the following

$$\begin{aligned}
t_0^{(i)} \| t_1^{(i)} \| t_2^{(i)} \| t_3^{(i)} &= S(k_{13}^{(i)} \| k_{14}^{(i)} \| k_{15}^{(i)} \| k_{12}^{(i)}) \\
k_0^{(i+1)} &= k_0^{(i)} + AB_i(t_0^{(i)}) + 2^{i-1} \\
k_j^{(i+1)} &= k_j^{(i)} + AB_i(t_j^{(i)}), \qquad (1 \le j < 4) \\
k_j^{(i+1)} &= k_j^{(i)} + k_{j-4}^{(i+1)}, \qquad (4 \le j < 16)
\end{aligned}$$

C Matrix and Round Constant Generation for Rain

For the sake of simplicity, in this section we omit using the round-specific indices of the linear layer matrices M_i and the linearized permutation polynomials L_i, i.e., we write M and L instead of M_i and L_i, respectively.

C.1 Building the Linear Layer M from $L(X)$

In order to build our linear layer M, we first pseudo-randomly generate a dense and linearized polynomial $L(X) \in \mathbb{F}_{2^n}[X]$ of maximum degree, i.e., a polynomial with n nonzero coefficients and a degree of 2^{n-1}. Note that we need this polynomial to be a permutation polynomial, and we also want the compositional inverse of this polynomial to fulfill the same properties.

For this purpose, we first define the Dickson matrix

$$D_L = \begin{pmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1}^2 & a_0^2 & \cdots & a_{n-2}^2 \\ \vdots & \vdots & \vdots & \vdots \\ a_1^{2^{n-1}} & a_2^{2^{n-1}} & \cdots & a_0^{2^{n-1}} \end{pmatrix}$$

of the linearized polynomial $L(X) = \sum_{i=0}^{n-1} a_i X^{2^i}$. It is known [LN96] that L is a permutation polynomial if and only if D_L is invertible.

In order to also ensure that the inverse of L fulfills the same properties (i.e., nonzero coefficients and maximum degree), we simply compute it. Indeed, the inverse $L^{-1}(X)$ of $L(X)$ is defined as

$$L^{-1}(X) = \frac{1}{\det(D_L)} \cdot \sum_{i=0}^{n-1} \overline{a}_i X^{2^i},$$

where $\overline{a}_i$ is the $(i,0)$-th cofactor of D_L. Hence, in order to generate a "good" polynomial $L(X)$, we pseudo-randomly generate coefficients $\{a_i\}_{i=0}^{n-1}$ until both $L(X)$ and $L^{-1}(X)$ have n nonzero coefficients and a degree of 2^{n-1}.

The second step is to generate the matrix $M \in \mathbb{F}_{2^n}^{n \times n}$ from L. This transformation has explicitly been shown in [Car63] and has been revisited in [WL13]. We quickly recall it here.

Assume $L(X)$ is given. Then, we first compute the ordered power basis

$$\{\beta_1, \beta_2, \ldots, \beta_n\} = \{\beta, \beta^2, \ldots, \beta^{n-1}\}$$

and its dual basis

$$\{\beta'_1, \beta'_2, \ldots, \beta'_n\},$$

i.e., a basis such that

$$\mathrm{tr}(\beta_i \beta'_j) = \delta_{i,j}$$

for $i \in [0, n]$ and $j \in [0, n]$, where $\mathrm{tr} : \mathbb{F}_{2^n} \to \mathbb{F}_2$ is the trace function and $\delta_{i,j}$ is the Kronecker delta. Now, the element $M_{i,j}$ at the j-th column of the i-th row in the matrix M is defined as

$$M_{i,j} = \mathrm{tr}(\beta'_i L(\beta_j)).$$

Hence, to summarize, we first generate a suitable linearized permutation polynomial $L(X)$, and we then compute the corresponding matrix. Ignoring the invertibility evaluation (with high probability, an invertible polynomial will be found after only a few trials), the cost of this approach is approximately $n^2 + n^3 \in \mathcal{O}(n^3)$ field operations. For example, using `Sage`, we are able to construct such a matrix for $n = 128$ in only a couple of seconds on an ordinary computer.

C.1.1 Pseudo-Randomly Sampling M

For completeness, we mention that it is also possible to construct the linear layer matrices by pseudo-randomly sampling them (i.e., without first choosing L and then computing the corresponding M). Here we argue that this approach will also lead to secure linear layers with high probability.

It is well-known that there exists a one-to-one relation between the general linear group $\mathrm{GL}(n, \mathbb{F}_2)$ and the set of linearized permutation polynomials with coefficients in $\mathbb{F}_{2^n}$, which is also called the Betti–Mathieu group [Dic01, Car63]. It is also known that the number of invertible $n \times n$ matrices over $\mathrm{GF}(2)$ is

$$\prod_{i=0}^{n-1} 2^n - 2^i > (2^n - 2^{n-1})^n = (2^{n-1})^n.$$

Since $n \log_2(2^{n-1}) = \log_2(2^{n^2}) - n$, it follows that at most n bits of entropy are lost when considering only invertible matrices. This means that, in the worst case, *no* maximum-degree monomial appears in the set of all linearized permutation polynomials of degree at most 2^{n-1} over a fixed field, and the appearances of all other degrees are uniformly distributed. Hence, the term with the second-highest degree will appear with an overwhelming probability of at least $1 - 2^{-n}$. However, in practical tests we observed that the probability of a maximum-degree term appearing is also around $1 - 2^{-n}$. Further, the probability of the polynomial to contain all powers of 2 less than or equal to 2^{n-1} is approximately $(1 - 2^{-n})^n$, as expected. We therefore conjecture that any pseudo-randomly chosen $n \times n$ matrix over $\mathbb{F}_2$ will result in a high-degree and dense (as far as possible) permutation over $\mathbb{F}_{2^n}$.

C.2 Pseudo-Random Number Generation

We generate the round constants and the linear layers using the SHAKE256 extendable output function [NIS15] with the input `Rain-N-R` (encoded as UTF-8 text), where N is replaced with the state size in bits and R is replaced with the number of rounds, e.g., `Rain-128-3` for $\textsc{Rain}_3$-128. We first generate the n-bit round constants $c^{(i)}$ by squeezing n bits from the SHAKE256 instance for each constant. Then, we use the same method to generate the coefficients for our linear layers.

C.3 Concrete Instances

The round constants and matrices used for $\textsc{Rain}$-n, where $n \in \{128, 192, 256\}$, are publicly available.[7] These files also contain the coefficients of the corresponding linearized permutation polynomials.

D Additional Performance Data and Evaluation

We explore the performance data of a huge range of instances for the variants presented in this paper at the 128-bit security level in Table 3. Most of these instances are included in Figure 9.

D.1 Instances for larger Security Levels

In Table 4, we give some numbers for Rainier for the 192-bit and 256-bit security levels.

We can see the same trends as for the 128-bit security level, where the LSAES-variants provide a 10-15% improvement in signature size over standard Banquet, while usually being about twice as fast. Recall that the EM construction is not directly instantiable with AES-192 and 256 since it requires an s-bit permutation for s-bit security. Rainier produces much smaller signatures that Banquet and also compares favorably to Picnic3 (with signatures of 27.4 KB at the 192-bit and 48.4 KB at the 256-bit security level) and SPHINCS$^+$ (with its "small" parameter sets having 17 KB at the 192-bit and 29.8 KB at the 256-bit security level, and its "fast" parameter sets being comparable in size to Picnic3). Furthermore, the signing and verification speeds of Rainier are again much faster than the other designs (with the exception of the verification times in SPHINCS$^+$, which are in the order of 0.5-5ms for all parameter sets). We give a visual representation of signing times and signatures sizes at the 192-bit and 256-bit security level for various schemes in Figures 10a and 10b.

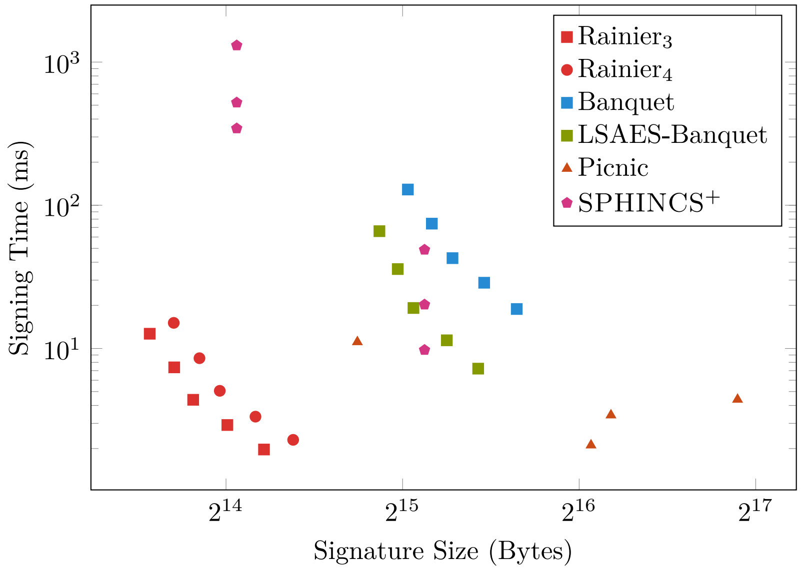
D.2 Alternative OWF Designs for MPCitH Signatures

We now take a quick look at further block ciphers that could be used in the proof protocol of Banquet [BdSGK$^+$21] or the modified variant presented in Section 5.2.2.

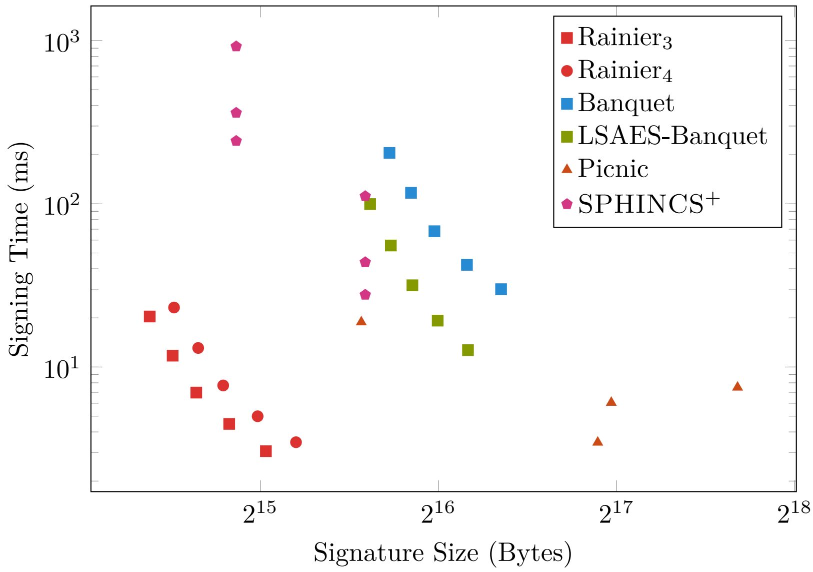
[7] `https://github.com/IAIK/rainier-signatures`

Design	N	τ	m_1	m_2	λ	Sign	Verify	Sig. size
Banquet	16	41	10	20	4	7.03	5.32	19 776
AES-128	31	35	10	20	4	10.01	8.27	17 456
	57	31	10	20	4	15.56	13.37	15 968
	107	24	10	20	6	23.03	20.45	14 784
	255	21	10	20	6	47.31	43.03	13 284
	512	20	10	20	6	90.30	82.67	12 976
	1024	18	10	20	6	161.51	148.55	11 976
Banquet	16	41	10	16	4	5.57	4.28	17 480
EM-AES-128	31	35	10	16	4	8.04	6.72	15 496
	57	31	10	16	4	12.54	10.89	14 232
	107	24	10	16	6	19.69	17.31	13 248
	255	21	10	16	6	41.05	36.88	11 940
	512	20	10	16	6	78.66	71.26	11 696
	1024	18	10	16	6	140.72	127.89	10 824
Banquet	16	41	5	10	1	2.93	2.37	16 496
LSAES-128	31	35	5	10	1	4.35	3.74	14 656
	64	31	5	10	1	7.49	6.54	13 488
	128	28	5	10	1	13.27	11.82	12 640
	256	25	5	10	1	25.28	22.92	11 696
	512	24	5	10	1	51.52	47.10	11 616
	1024	22	5	10	1	98.97	91.77	11 008
Banquet	16	41	5	8	1	2.50	1.99	14 528
EM-LSAES-128	31	35	5	8	1	3.71	3.15	12 976
	64	31	5	8	1	6.30	5.47	12 000
	128	28	5	8	1	11.48	10.16	11 296
	256	25	5	8	1	20.99	18.91	10 496
	512	24	5	8	1	40.66	36.47	10 464
	1024	22	5	8	1	79.21	72.57	9 952
Rainier$_3$	8	44	-	-	-	0.70	0.60	10 656
RAIN$_3$-128	16	33	-	-	-	0.87	0.81	8 544
	31	27	-	-	-	1.29	1.23	7 440
	57	23	-	-	-	1.87	1.82	6 720
	107	20	-	-	-	2.96	2.92	6 176
	256	17	-	-	-	5.65	5.63	5 536
	920	14	-	-	-	16.82	16.70	5 024
	1624	13	-	-	-	28.28	28.16	4 880
	3180	12	-	-	-	51.90	51.49	4 704
	7121	11	-	-	-	105.98	105.15	4 496
	65384	9	-	-	-	801.98	794.35	4 128
Rainier$_4$	8	44	-	-	-	0.81	0.71	12 064
RAIN$_4$-128	16	33	-	-	-	1.03	0.96	9 600
	31	27	-	-	-	1.50	1.44	8 304
	57	23	-	-	-	2.19	2.14	7 456
	107	20	-	-	-	3.47	3.42	6 816
	256	17	-	-	-	6.65	6.60	6 080
	920	14	-	-	-	20.01	19.76	5 472
	1625	13	-	-	-	33.41	33.11	5 296
	3181	12	-	-	-	60.88	60.49	5 088
	7124	11	-	-	-	124.88	123.79	4 848
	65422	9	-	-	-	952.73	941.02	4 416

Table 3: Performance comparison of signatures discussed in this work at the 128-bit security level. Times are in ms, sizes are in bytes.



(a) 192-bit security level.



(b) 256-bit security level.

Figure 10: Comparison of signing time and signature size of various schemes at 192-bit and 256-bit security levels. Picnic instances include all proposed third round parameter sets. SPHINCS$^+$ instances include all `simple` parameter sets (`haraka`,`sha256`,`shake256`, in order of decreasing performance).

Vision [AAB$^+$20]. *Vision* is a block cipher design intended for use-cases in multi-party computation and zero-knowledge proofs with the goal of reducing the arithmetic complexity of the cipher. *Vision* is a family of block ciphers operating over binary fields, where the only non-linear operation is the field inverse. This makes *Vision* an easy fit for the Banquet proof protocol. One point of remark is *Vision*'s heavy key schedule, which essentially corresponds to a second evaluation of the cipher itself; while in many scenarios this can be amortized over many encryption calls, for the signature use-case we only prove a single encryption, resulting in a large overhead. However, we can also use *Vision* in a single-key Even–Mansour construction as discussed in Section 2.1. We also give the sizes for these variants. While we have not implemented the full signature for the following variants, we give the calculated signature sizes for three different *Vision* instances at the 128-bit security level in the Banquet proof protocol in Table 5. The first one is *Vision Mark I*, an instance recommended in [AAB+20] intended to be similar to AES. The second and third ones are generated with the provided parameter generation script[8] intended to resemble the design choices of LSAES and Rain with field sizes of 32 and 128 bits respectively. However, we remark that the resulting parameters are for general use and could potentially be reduced for the attack scenario presented by the use in the signature construction, where only a single (plaintext, ciphertext) pair is published.

Table 5 highlights our remarks about the heavy key schedule of *Vision* as the signatures produced by using Vision instances are larger than all other alternatives we investigated and even unmodified Banquet. When considering the EM variants, we see that the signature size of EM-Vision Mark I and EM-Vision Variant 2 are equal to EM-AES-128 and EM-LSAES-128, respectively. This is because of the similarities of the internal structure of the ciphers, resulting in the same number of inversions for the same field sizes. For Variant 3, the number of internal rounds is much larger than Rain, since the parameter selection script takes into account all types of attacks, even those not applicable in the signature scenario. A future dedicated analysis could reduce the number of rounds needed for security, but due to the structure in the affine layer of *Vision* the number of rounds will very likely need to be larger than for Rain.

LegRoast and PorcRoast [Bd20]. Leg- and PorcRoast are two recent proposals of MPCitH-based signature schemes that use the Legendre PRF instead of a block cipher as a one-way function. Security is based on a more structured number-theoretic assumption, which arguably puts these in a different category than the schemes in this paper, but we provide a brief comparison. On our machine, the LegRoast parameter sets have signature sizes of 12.5 to 16.48 KB with a signing time of 3.19 to 17.95 ms, while the PorcRoast parameter sets have signature sizes of 6.4 to 8.8 KB with signing times of 1.43 to 8.85 ms (with larger signatures leading to faster signing times). Public keys are 4 KB, compared to 32 bytes in Rainier.

LSAES with Larger S-Boxes. As mentioned above, the choice of 32-bit S-boxes in LSAES was made to match the field $\mathbb{F}_{2^{8\lambda}}$ used in Banquet (where $\lambda = 4$ for L1). However, we can further reduce the number of S-boxes by using 64-bit or even 128-bit S-boxes. Since the (LS)AES key schedule operates on 32-bits at a time, this is the largest size for the key schedule, but rather than mixing S-box sizes, here we focus on EM-LSAES and for simplicity fix $N = 256$. With 64-bit S-boxes, EM-LSAES has only 20 S-boxes, and the signature size

[8] https://github.com/KULeuven-COSIC/Marvellous

is 9 168 bytes using the Rainier protocol and 9 216 bytes using the Banquet protocol, compared to 10 496 bytes for 32-bit S-boxes (Table 3). With 128-bit S-boxes, EM-LSASES has only 10 S-boxes and sigatures are 9 072 or 9 312 bytes using the simplified or full Banquet protocol. Unfortunately sizes do not really decrease further with larger S-boxes, and Rain gives much better performance. This can be explained by having 10 rounds in the LSAES variants vs. 3 rounds in Rain. Indeed, note that signature sizes of r-round EM-LSAES with 128-bit S-boxes is the same as r-round Rain. This motivates study of reduced-round versions of EM-LSAES, which has a weaker linear layer than Rain, however, we expect that 7 and perhaps even 5 rounds to be sufficient.

Other AES-Based Permutations. In order to use the EM OWF construction at higher security levels, we require 192 and 256-bit permuataions. Using Rijndael is one option, but there are also constructions of permuataions from AES. The motivation for these constructions is that they make use of hardware aceleration for AES (such as the AESNI instructions), while no such hardware is available for Rijndael.

The Simpira v2 permuatation (with $b \geq 2$) together with the EM construction gives a candidate OWF for Banquet-like signatures at security level L5 (256-bit security) [GM16]. However, the number of AES rounds required is 30 rounds, compared to AES-256x2 (two calls to AES-256), which uses 28 AES rounds + the key schedule. So it seems Simpira v2 is not a good choice for our application.

Haraka v2 [KLMR16] defines the permutation π_{256}. This permutation has 5 rounds, where each π_{256} round uses four AES rounds, for a total of 20 AES rounds. Aside from the AES rounds, the state is permuted, so counting the AES rounds is sufficient when discussing signature signature sizes. The parameters in [KLMR16] were chosen and analyzed to provide 256 bits of second preimage resistance, meaning Haraka-v2-256 is a OWF suitable for use in a Banquet-like scheme, where key generation is $pk = \text{Haraka-v2}(sk) = sk \oplus \pi_{256}(sk)$ for random 256-bit secret key sk (ignoring that a salt is required to prevent multi-target attacks; we assume some of the round constants can be selected per-party, to give each user a distinct instance of π_{256}).

Then Banquet-Haraka-v2-256 provides 256-bit security, with 320 eight-bit S-boxes. Signatures range from 66.7–44.4 KB (with N =16–256) better than any of the Banquet-based options in Table 4, but worse than the Rainier options.

One can also consider the Haraka v2 analogues constructed with LSAES rounds, instead of AES rounds (with the caveat that the security analysis must be revisited). Then instead of 320 8-bit S-boxes, we have 80 32-bit S-boxes. Here the signature sizes range from 54.6–39.4 KB (with N =16–256).

Design	N	τ	m_1	m_2	λ	Sign	Verify	Sig. size
Banquet AES-192x2	16	62	16	26	4	18.86	14.24	51 216
	32	53	16	26	4	28.77	23.95	45 072
	64	40	16	26	6	42.72	36.98	39 808
	128	36	16	26	6	74.43	66.18	36 704
	256	32	16	26	6	128.80	116.41	33 408
Banquet AES-256x2	16	84	20	25	4	29.99	23.23	83 488
	32	63	20	25	6	42.30	35.23	73 114
	64	54	20	25	6	67.96	59.51	64 420
	128	48	20	25	6	116.85	104.65	58 816
	256	43	20	25	6	205.26	185.95	54 082
Banquet LSAES-192x2	16	62	8	13	1	7.23	5.81	44 024
	32	53	8	13	1	11.40	9.65	38 924
	64	45	8	13	1	19.17	16.68	34 148
	128	41	8	13	1	35.85	32.04	32 108
	256	37	8	13	1	65.92	59.43	29 876
Banquet LSAES-256x2	16	84	5	25	1	12.69	10.00	73 408
	32	72	5	25	1	19.25	15.88	65 248
	64	63	5	25	1	31.69	26.91	59 128
	128	56	5	25	1	55.57	48.15	54 368
	256	50	5	25	1	99.71	88.17	50 160
Rainier$_3$ Rain$_3$-192	16	49	-	-	-	1.97	1.77	18 944
	32	40	-	-	-	2.90	2.76	16 448
	64	33	-	-	-	4.38	4.23	14 384
	128	29	-	-	-	7.36	7.22	13 352
	256	25	-	-	-	12.85	12.60	12 128
Rainier$_4$ Rain$_4$-192	16	49	-	-	-	2.30	2.07	21 296
	32	40	-	-	-	3.34	3.15	18 368
	64	33	-	-	-	5.06	4.90	15 968
	128	29	-	-	-	8.55	8.38	14 744
	256	25	-	-	-	15.09	14.81	13 328
Rainier$_3$ Rain$_3$-256	16	65	-	-	-	3.05	2.83	33 440
	32	53	-	-	-	4.48	4.27	28 992
	64	44	-	-	-	6.97	6.80	25 504
	128	38	-	-	-	11.74	11.57	23 264
	256	33	-	-	-	20.40	20.23	21 280
Rainier$_4$ Rain$_4$-256	16	65	-	-	-	3.46	3.18	37 600
	32	53	-	-	-	4.99	4.75	32 384
	64	44	-	-	-	7.71	7.47	28 320
	128	38	-	-	-	13.08	12.80	25 696
	256	33	-	-	-	23.16	22.73	23 392

Table 4: Performance comparison at the 192-bit and 256-bit security levels for $N \in \{16, 32, 64, 128, 256\}$. Times are in ms, sizes are in bytes.

Instance	κ	n	m	r	**#inv.**	**Sig. size**
Vision Mark I	128	8	16	10	320	21176
Vision Variant 2	128	32	4	10	80	17952
Vision Variant 3	128	128	1	36	72	41184
EM-Vision Mark I	128	8	16	10	160	14232
EM-Vision Variant 2	128	32	4	10	40	12000
EM-Vision Variant 3	128	128	1	36	36	25056
Banquet (AES-128)	128	8	16	10	200	15968
Banquet (EM-AES-128)	128	8	16	10	160	14232
Banquet (LSAES-128)	128	32	4	10	50	13488
Banquet (EM-LSAES-128)	128	32	4	10	40	12000
Rainier	128	128	1	3	3	6720
Rainier$_{\mathrm{R}}$	128	128	1	4	4	7456

Table 5: Signature sizes in bytes for different instances of Vision with a security level κ using a state of m elements over the field $\mathbb{F}_{2^n}$ and r rounds of the cipher (resulting in a total of $\#inv.$ inverses). The proof system parameters are set to $N = 64$ with the minimum number of rounds τ chosen so that the signature scheme provides κ bits of security and the remaining parameters chosen as recommended by [BdSGK+21].