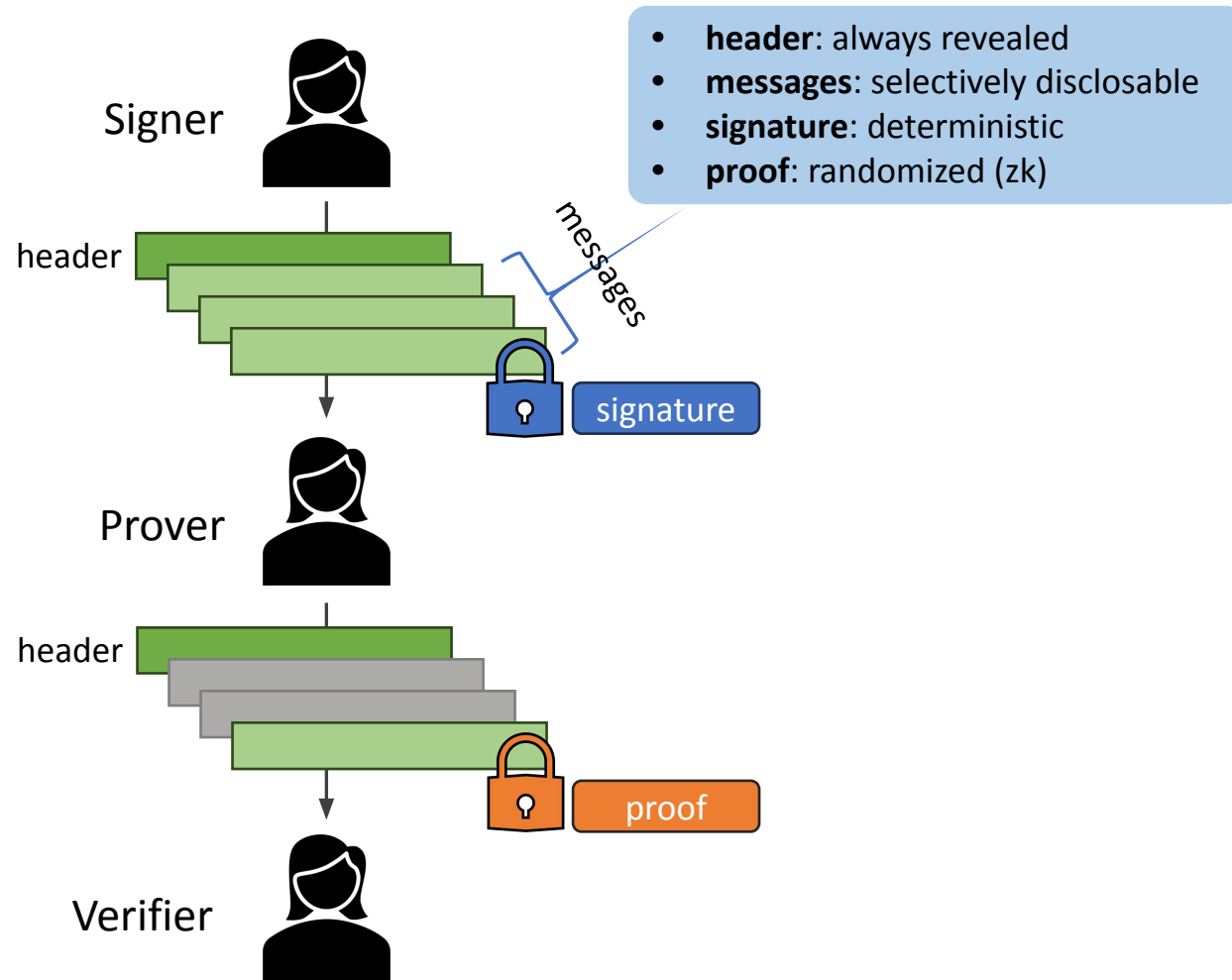


The BBS Signature Scheme

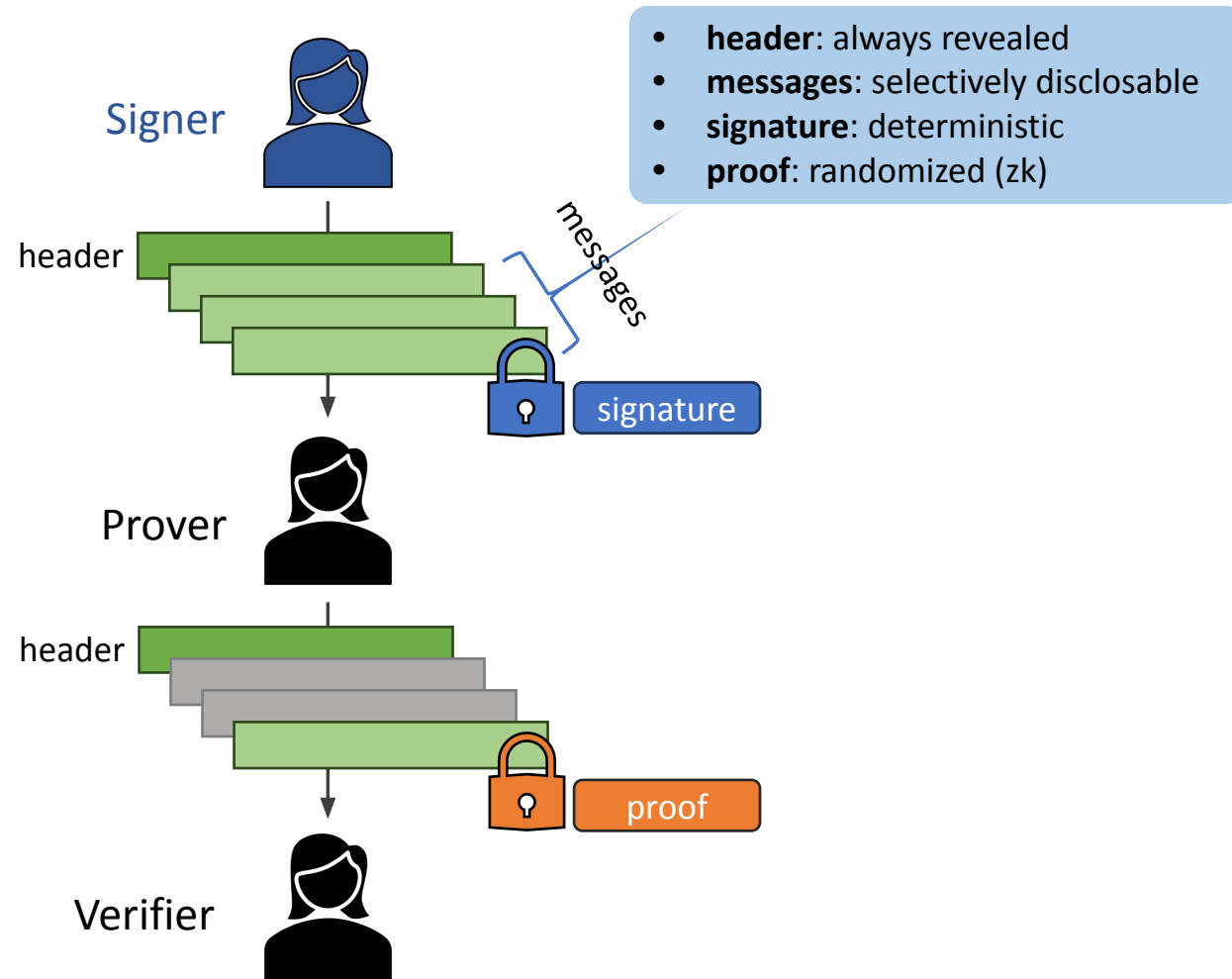
Tobias Looker, Vasilis Kalos, Andrew Whitehead, Mike Lodder

BBS Signatures Recap



Multi message signature, supporting ZK-Proofs while selectively disclosing the messages.

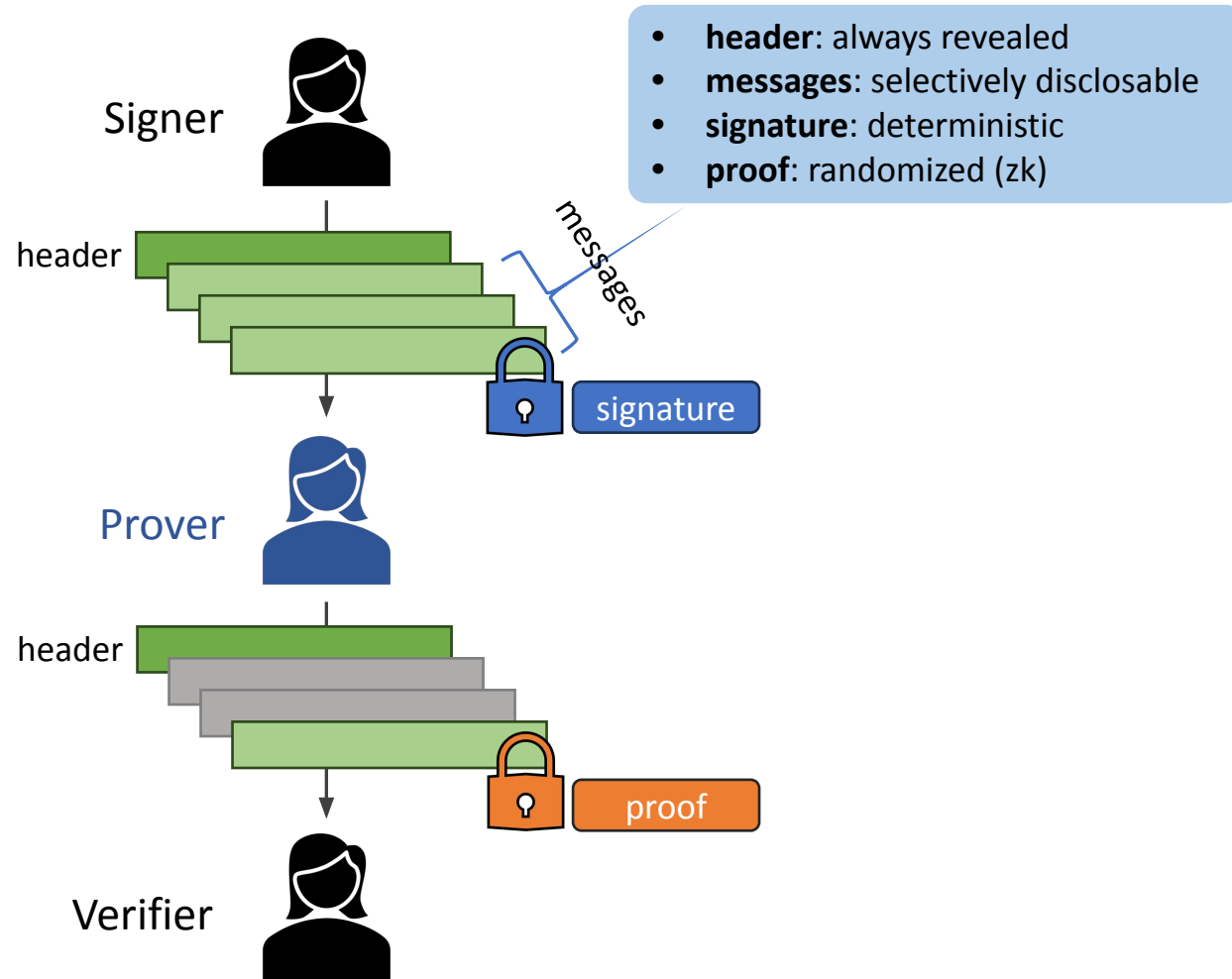
BBS Signatures Recap



Signer:

- Sign multiple messages + header

BBS Signatures Recap



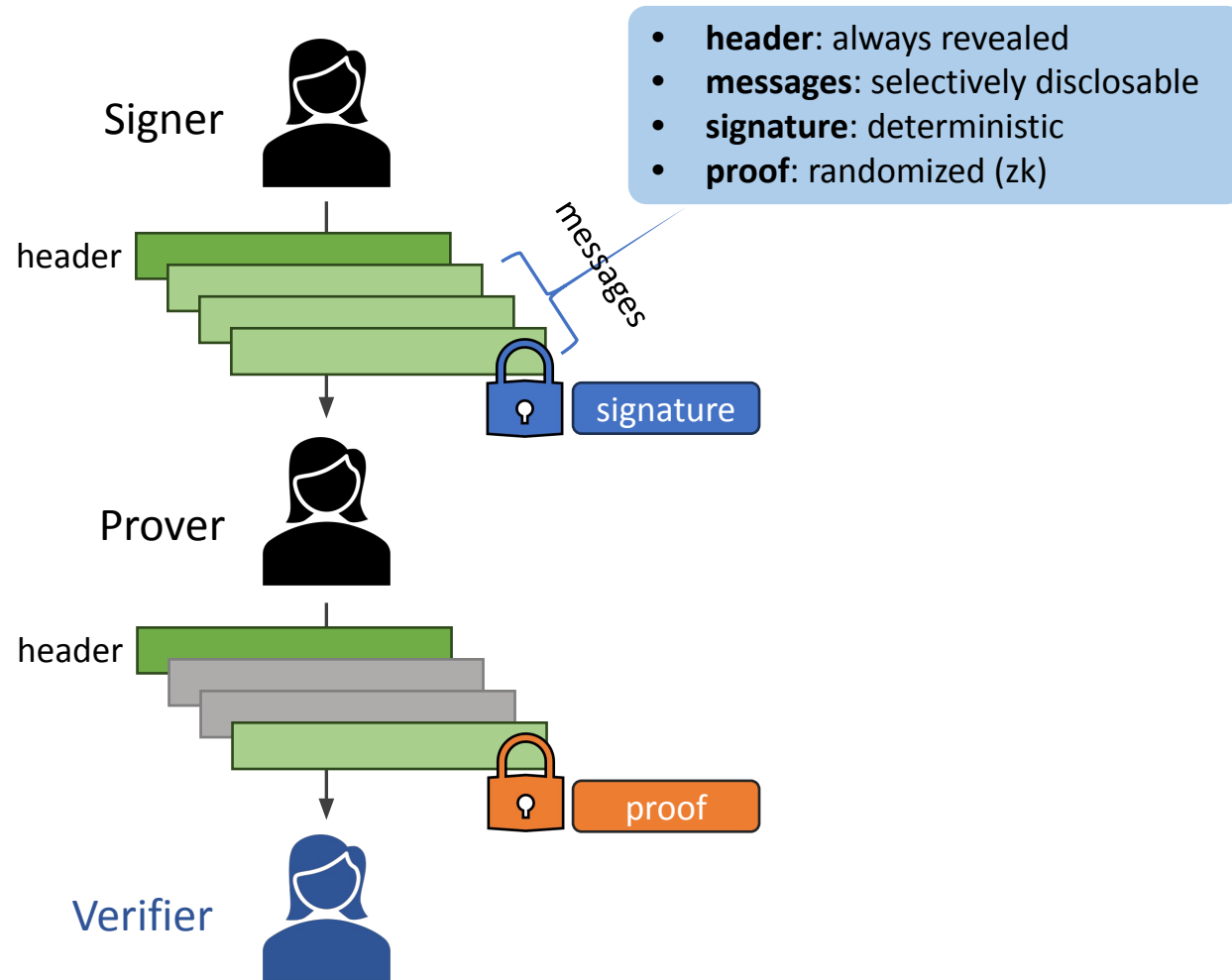
Signer:

- Sign multiple messages + header

Prover:

- Verify Signature
- Choose messages to disclose
- Optionally choose a presentation header
 - Context to be bound to the proof (e.g., a nonce, a date etc)
- Generate a ZK-Proof

BBS Signatures Recap



Signer:

- Sign multiple messages + header

Prover:

- Verify Signature
- Choose messages to disclose
- Optionally choose a presentation header
 - Context to be bound to the proof (e.g., a nonce, a date etc)
- Generate a ZK-Proof

Verifier:

- Verify proof on revealed messages
- Guarantees integrity and authenticity of the revealed messages + knowledge of a BBS signature created by the Signer

BBS Updates: News from Accademia

Revisiting BBS Signatures*

Stefano Tessaro  and Chenzhi Zhu 

Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, US
{tessaro,zhucz20}@cs.washington.edu

Abstract. BBS signatures were implicitly proposed by Boneh, Boyen, and Shacham (CRYPTO '04) as part of their group signature scheme, and explicitly cast as stand-alone signatures by Camenisch and Lysyanskaya (CRYPTO '04). A provably secure version, called BBS+, was then devised by Au, Susilo, and Mu (SCN '06), and is currently the object of a standardization effort which has led to a recent RFC draft. BBS+ signatures are suitable for use within anonymous credential and DAA systems, as their algebraic structure enables efficient proofs of knowledge of message-signature pairs that support partial disclosure.

BBS+ signatures consist of one group element and two scalars. As our first contribution, we prove that a variant of BBS+ producing shorter signatures, consisting only of one group element and one scalar, is also secure. The resulting scheme is essentially the original BBS proposal, which was lacking a proof of security. Here we show it satisfies, under the q -SDH assumption, the same provable security guarantees as BBS+. We also provide a complementary tight analysis in the algebraic group model, which heuristically justifies instantiations with potentially shorter signatures.

Furthermore, we devise simplified and shorter zero-knowledge proofs of knowledge of a BBS message-signature pair that support partial disclosure of the message. Over the BLS12-381 curve, our proofs are 896 bits shorter than the prior proposal by Camenisch, Drijvers, and Lehmann (TRUST '16), which is also adopted by the RFC draft.

Finally, we show that BBS satisfies one-more unforgeability in the algebraic group model in a scenario, arising in the context of credentials, where the signer can be asked to sign arbitrary group elements, meant to be commitments, without seeing their openings.

1 Introduction

The seminal works of Camenisch and Lysyanskaya [CL03, CL04] highlighted how certain digital signature schemes with suitable algebraic structures are amenable to applications such as anonymous credentials, direct anonymous attestation (DAA), and group signatures. These schemes easily enable the signing of a *commitment*, typically by being algebraically compatible with a Pedersen commitment [Ped92], and support very efficient zero-knowledge proofs of knowledge of a valid message-signature pair.

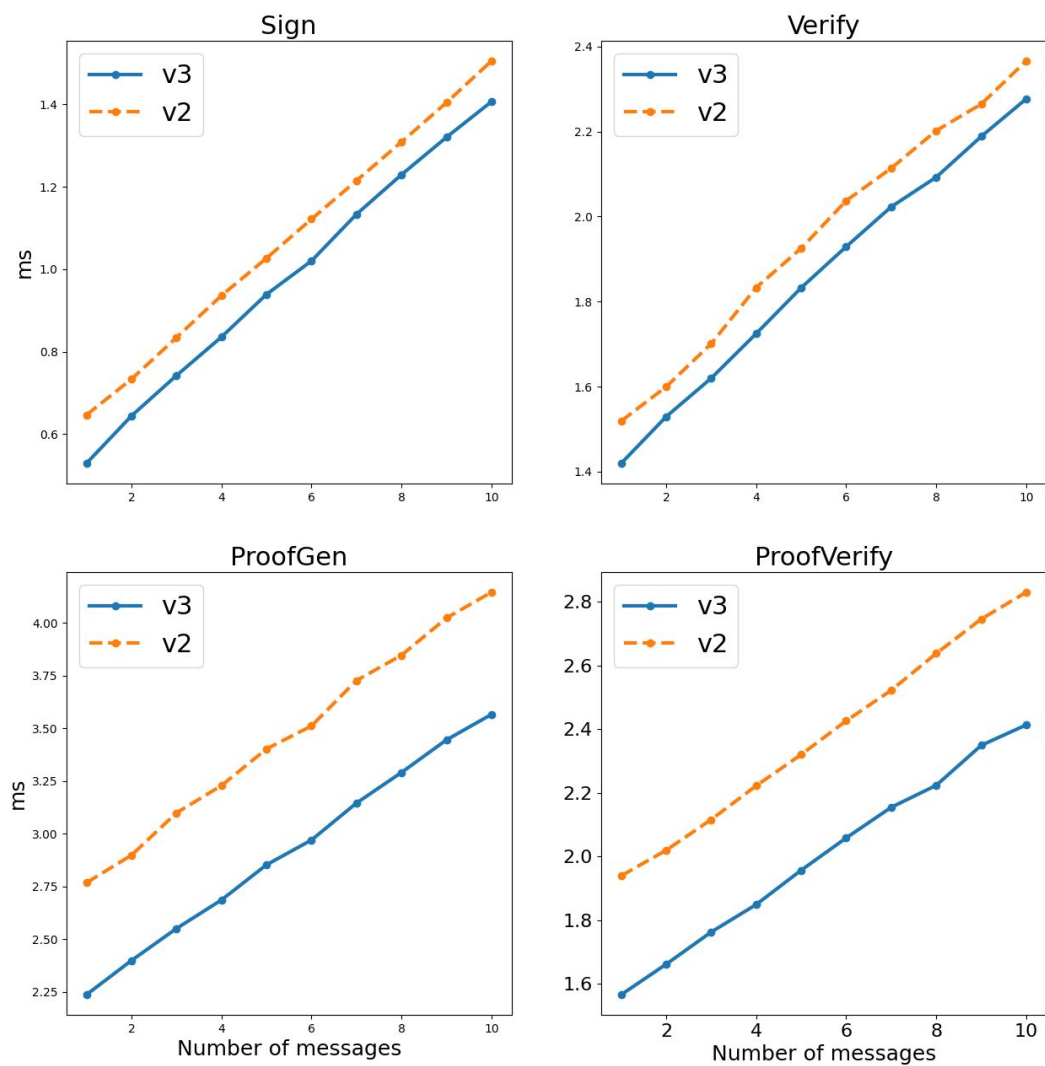
This paper revisits and improves BBS signatures [BBS04, ASM06, CDL16], one of the most efficient pairing-based schemes with these properties, which has recently been in the midst of renewed interest in the context of decentralized identity. This has led to reference implementations [BBSa, BBSb], to a standardization effort by the W3C Verifiable Credentials Working group, and to an RFC draft [LKWL22]. BBS is also a building block for DAA [Che09, BL10, CDL16], and is used by Intel SGX's EPID protocol [BL11]. Furthermore, BBS signatures are theoretically

New paper in EUROCRYPT2023 revisiting BBS [1].

Improved efficiency and signature/ proof sizes

- Signature size decrease by 32 bytes
 - 112 → 80 bytes
- Proof size decrease by 112 bytes
 - $304 + L * 32 \rightarrow 192 + L * 32$ bytes

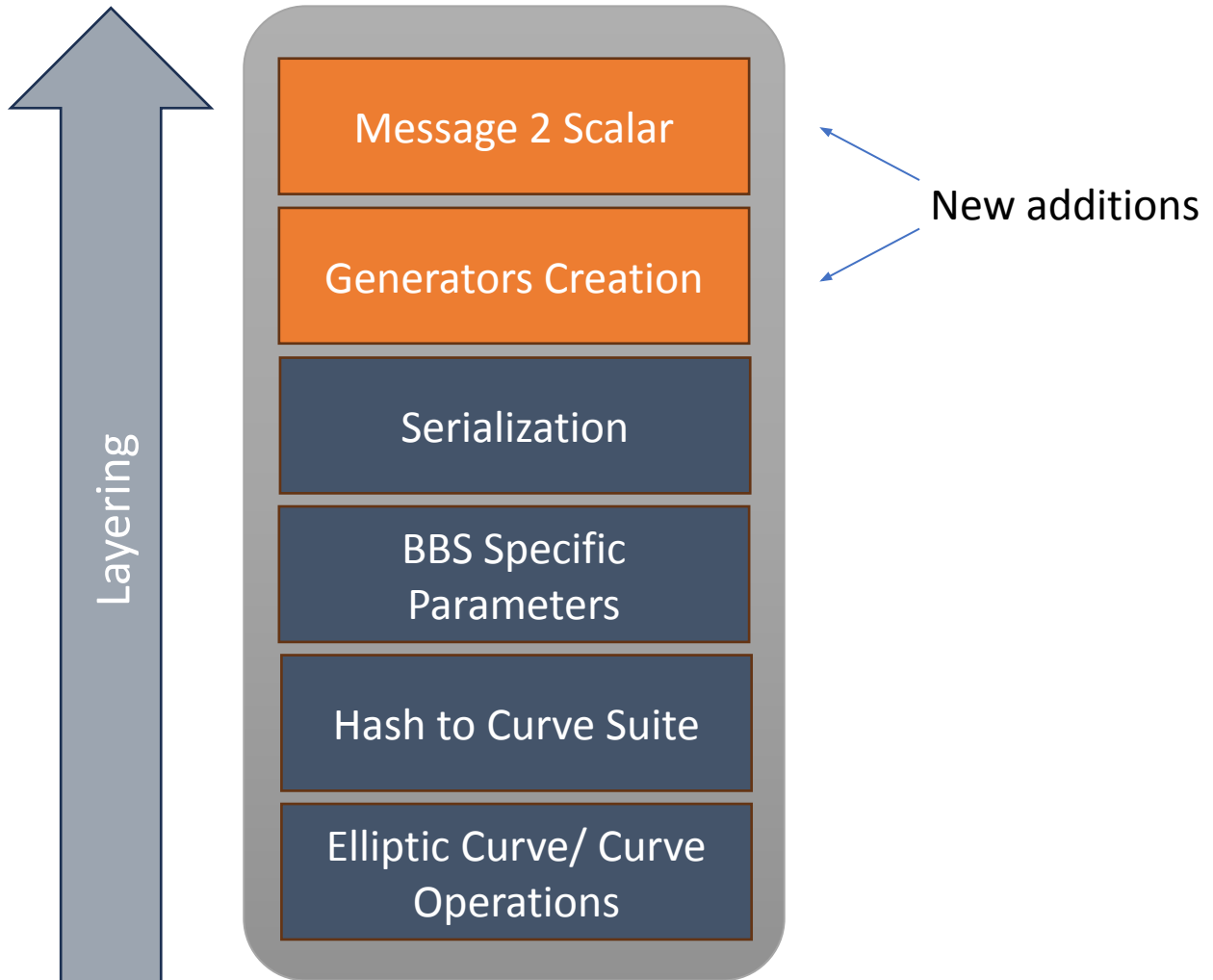
Performance Improvements



Performance difference between draft-02 and draft-03

- Improved performance by 20% in **Sign**, 7% in **Verify**, 50% in **ProofGen** and 25% in **ProofVerify**

BBS Updates: Ciphersuite Additions



BBS Ciphersuite

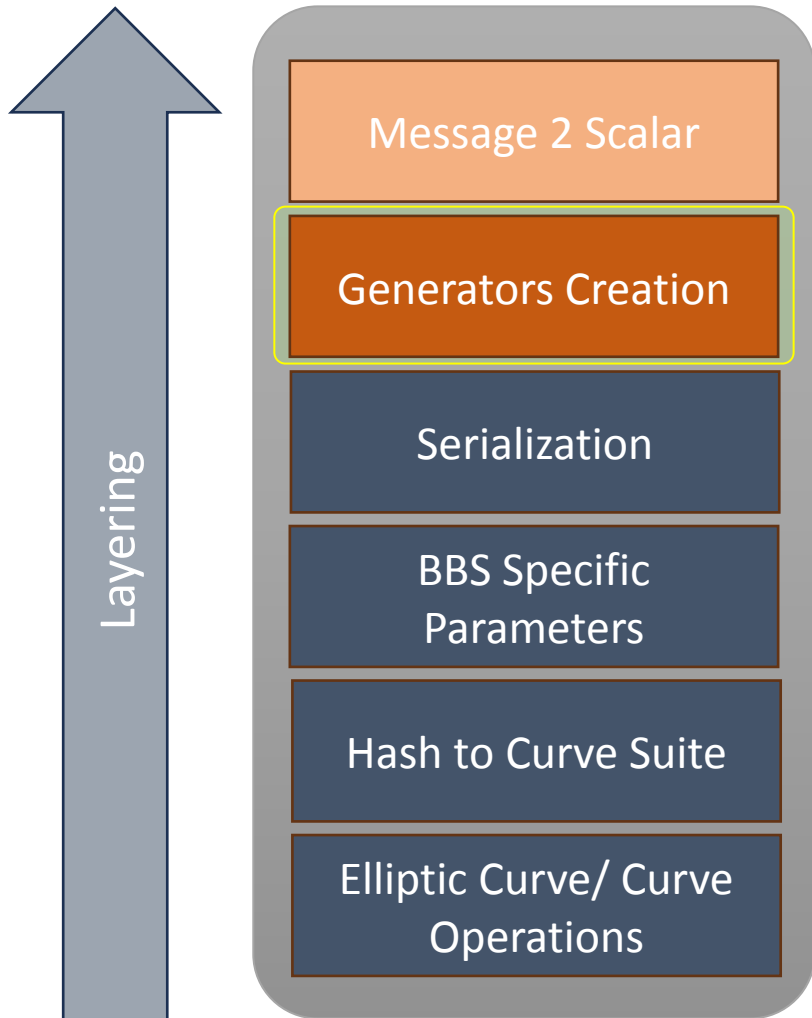
Build with:

- A pairing friendly elliptic curve
- A Hash-to-Curve suite
- Some BBS Specific parameters
- Serialization functions

New Additions:

- A generators creation operation
- A message to scalar mapping operation

BBS Updates : Generators Flexibility



Create Generators Procedure

Generators:

- random, independent, distinct EC points
- Need one generator per message

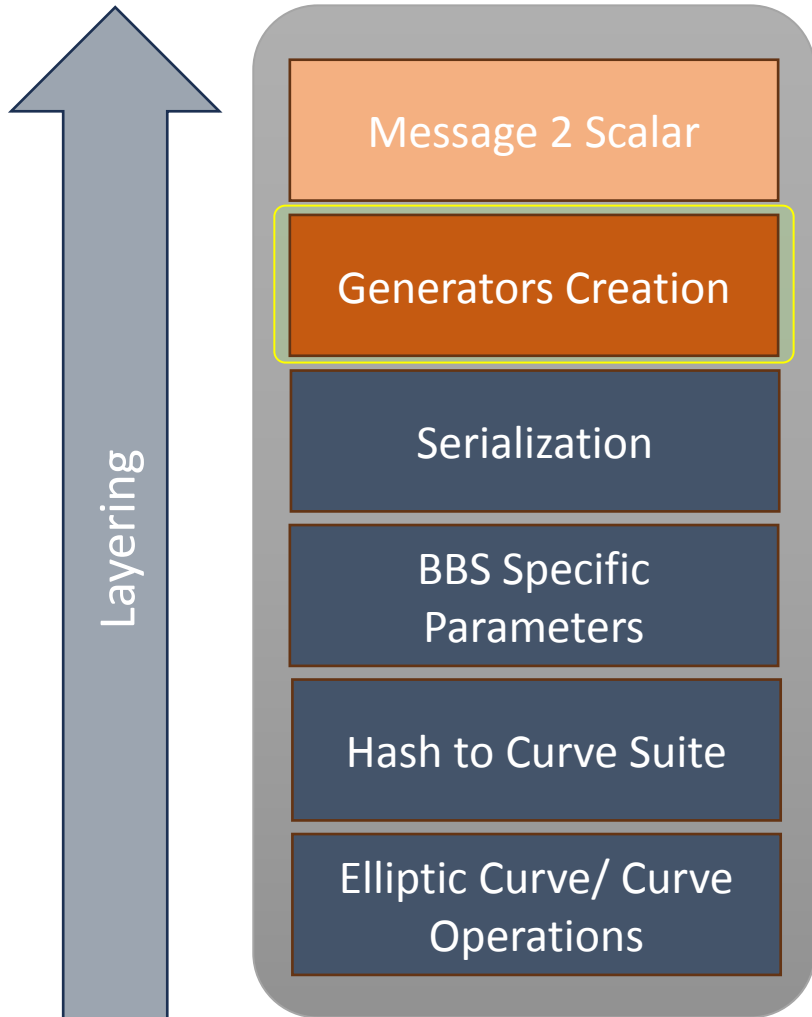
Current Procedure:

- Using `hash_to_curve` (in “feedback + counter mode”)
- Easily extent a set of already computed generators

Need for flexibility examples:

- Holder Binding -> use a distinguished point as a generator
- Blind Signatures -> use a set of pre-defined generators for the blinded messages

BBS Updates : Generators Flexibility

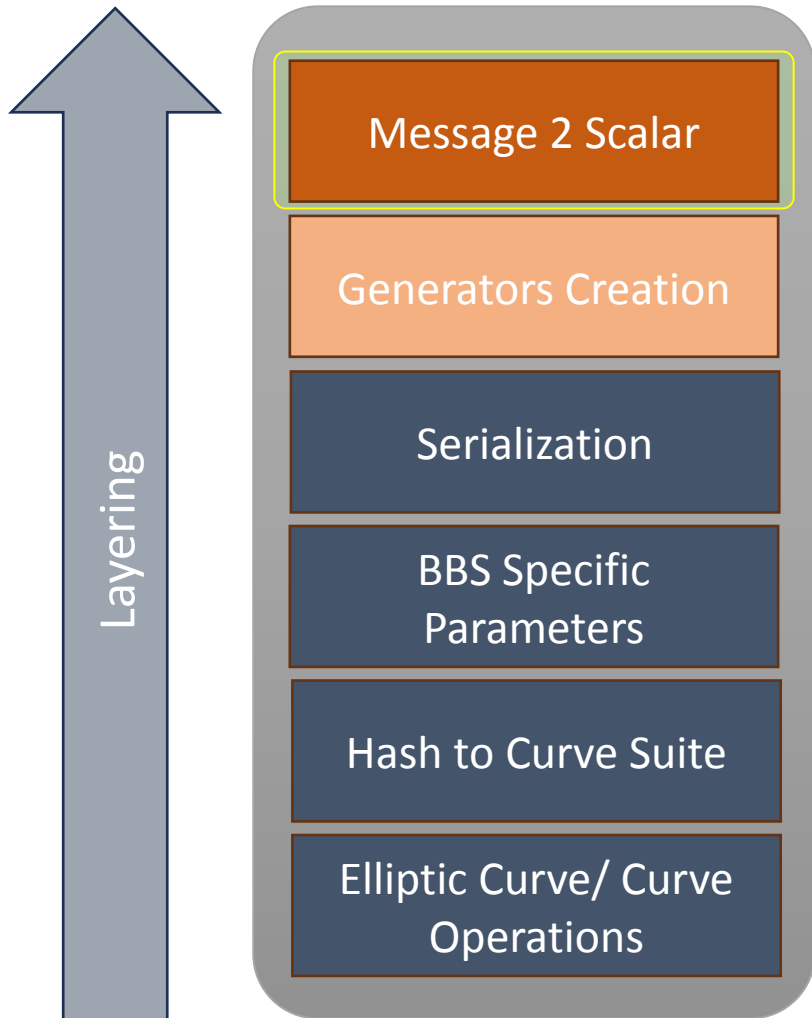


Create Generators Procedure

Allowed Customization:

- Allow ciphersuites to define new generators creation method.
- Optionally, return Issuer specific generators (the ones currently defined are global).

BBS Updates: Messages to Scalars



Message to Scalar Procedure

Messages need to be mapped to scalars before signed

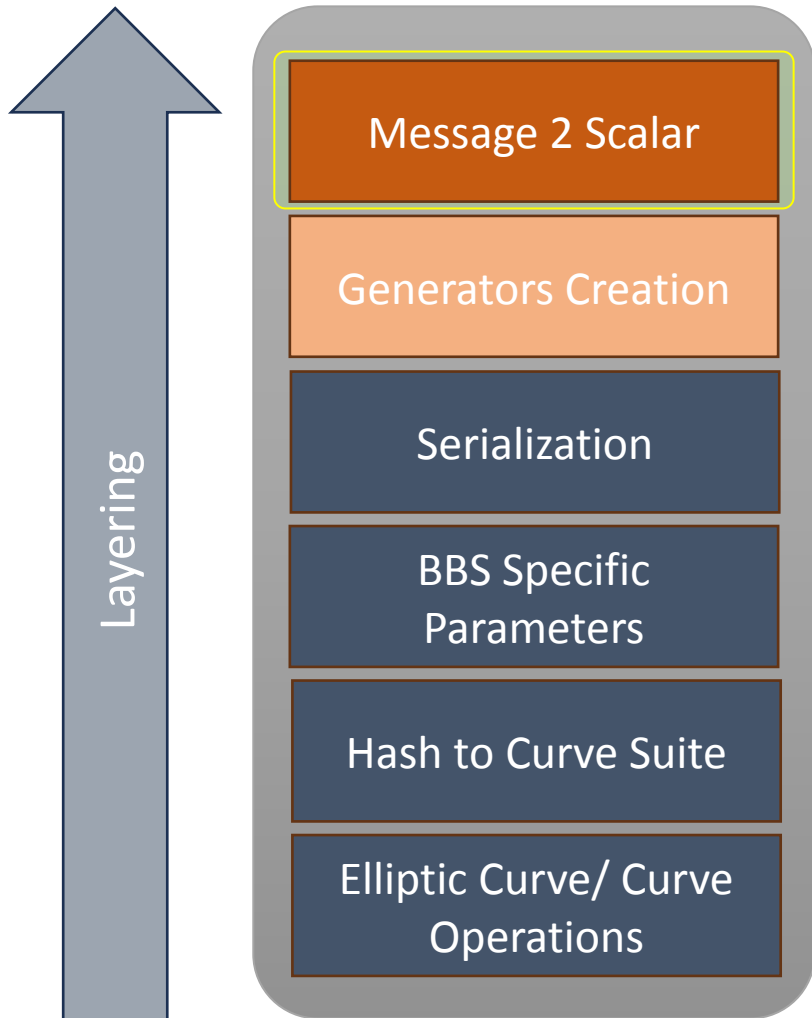
Current Procedure:

- Use hash-to-scalar (based on expand-message from h2c) to map octet strings into scalars
- Only accept octet strings as messages

Need for flexibility examples:

- Holder Binding -> one of the messages will be already a scalar (a sk belonging to the user).
- Predicate Proofs -> range proofs on signed scalars.

BBS Updates: Messages to Scalars

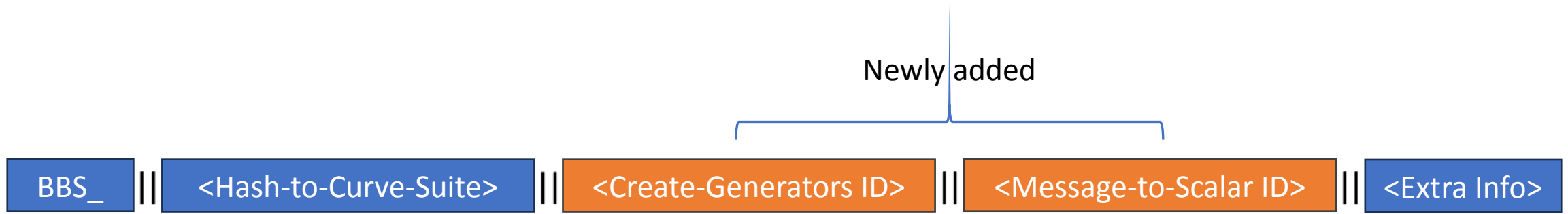


Message to Scalar Procedure

Allowed Customization:

- Allow ciphersuites to define new mappings, giving the ability to directly sign scalars.

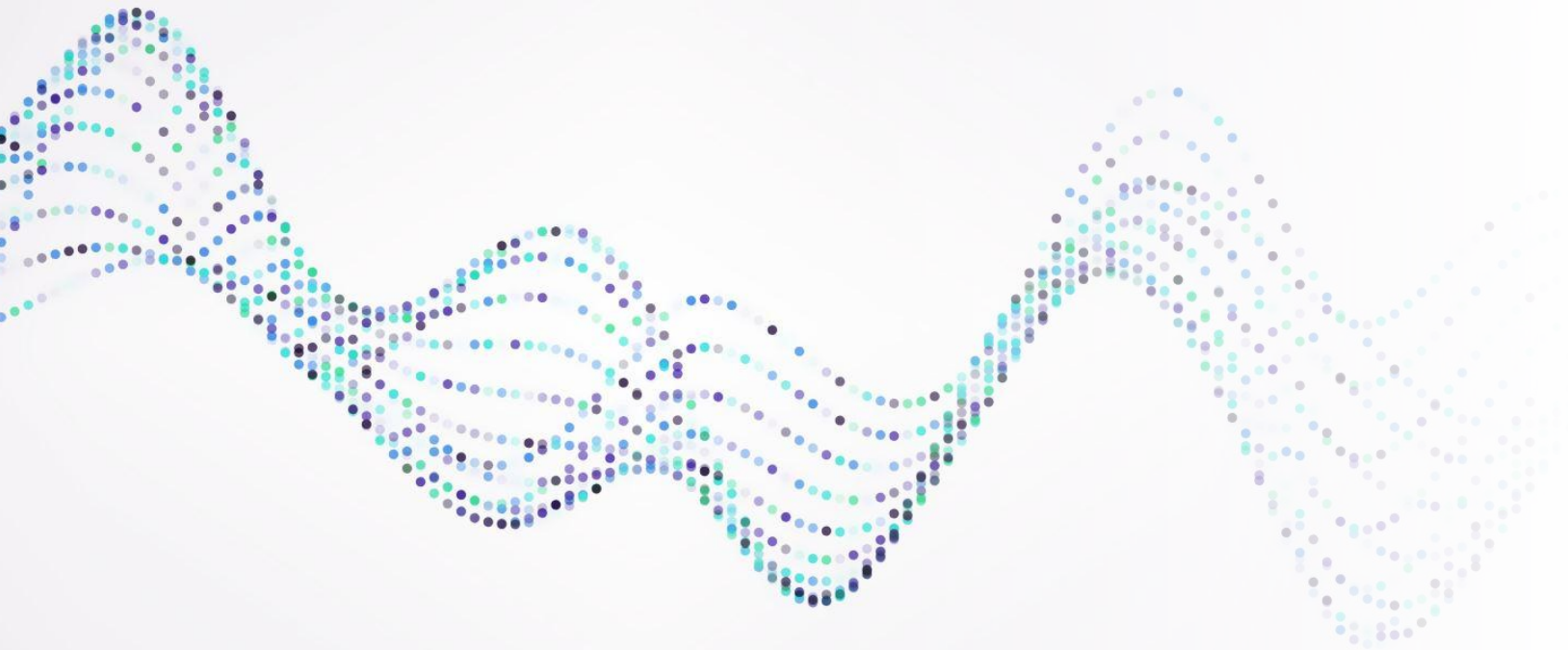
New Ciphersuite ID



- The generation-creation and message-to-scalar mapping define a unique ID
- Add those IDs to the Ciphersuite ID

Some More Updates

1. Kept test vectors on the core document for know. Will consider moving some of them.
2. Added new test vectors (using variable length messages, no-header/presentation header).
3. Updated Error handling (add the option to ABORT).
4. New KDF based on `hash_to_scalar` instead of HMAC (removed the need to directly use a hash).



Thank You!

Questions?