The BBS Signature Scheme

Tobias Looker, Vasilis Kalos, Andrew Whitehead, Mike Lodder
BBS Signatures Recap

- **header**: always revealed
- **messages**: selectively disclosable
- **signature**: deterministic
- **proof**: randomized (zk)

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

- **Signer**
  - Sign multiple messages + header

- **Prover**
  - **header**: always revealed
  - **messages**: selectively disclosable
  - **signature**: deterministic
  - **proof**: randomized (zk)

- **Verifier**
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:**

- **header:** always revealed
- **messages:** selectively disclosable
- **signature:** deterministic
- **proof:** randomized (zk)
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

- **header:** always revealed
- **messages:** selectively disclosable
- **signature:** deterministic
- **proof:** randomized (zk)
BBS Updates: News from Accademia

Revisiting BBS Signatures

Stefano Tessaro and Chenhui Zhu

Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, WA
(tessaro,shuzhui@cs.washington.edu)

Abstract. BBS signatures were implicitly proposed by Benaloh, Bayer, and Chaum (CRYPTO'94) as part of their group signature scheme, and explicitly as stand-alone signatures by Camenisch and Lysyanskaya (CRYPTO'01). A practically secure version, called BBS+, was then devised by Au, Balsam, and Ma (NC'08), 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 secrets. As our first contribution, we prove that a variant of BBS+ providing shorter signatures, consisting only of one group element and one scalar, is also secure. The resulting scheme is essentially the original BBS proposal, which lacked 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. Once the 32-bit 32-byte curve, our proofs are 98 bits shorter than the prior proposal by Camenisch, Drijvers, and Lohmann (TRUST'14), which is also adopted by the RFC draft.

Finally, we show that BBS suffers one more unfeasibility 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 [CL04, 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 [Ped91], and support very efficient zero-knowledge proofs of knowledge of a valid message-signature pair.

This paper revisits and improves BBS signatures [HBB04, AS06, CL16], 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 new implementation improvements (BBS+, IBBS), in a standardization effort by the W3C Verifiable Credentials Working group, and to an RFC draft [LKW12]. BBS is also a building block for DAA [Che06, BL10, CL16], and it is used by both ISO’s EPID protocol [BL14]. Furthermore, BBS signatures are theoretically

New paper in EUROCRYPT2023 revisiting BBS [1].

Improved efficiency and signature/ proof sizes
- Signature size decrease by 32 bytes
  - $112 \rightarrow 80$ bytes
- Proof size decrease by 112 bytes
  - $304 + L \times 32 \rightarrow 192 + L \times 32$ bytes

Performance Improvements

- 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

Layering:
- Message 2 Scalar
- Generators Creation
- Serialization
- BBS Specific Parameters
- Hash to Curve Suite
- Elliptic Curve/Curve Operations

New additions
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 extend 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?