BBS Signature Suite Advanced Features

Blind Signing and Pseudonyms

Vasilis Kalos and Greg Bernstein

Building on The BBS Signature Scheme

Blind BBS Signatures

https://www.ietf.org/archive/id/draft-kalos-bbs-blind-signatures-03.html

Provides for selective disclosure and unlinkable proofs over *committed* messages from **prover** as well as the messages from **signer**

BBS per Verifier Linkability

https://www.ietf.org/archive/id/draft-kalos-bbs-per-verifier-linkability-00.html

Provides for “controlled/limited” linkability when a **prover** revisits a **verifier** or group of verifiers while maintaining general unlinkability and untraceability.

Same three party model as The BBS Signature Scheme

- **Signer:** Signs an ordered set of messages (statements)
- **Prover:** Chooses a subset of messages to disclose and provides a BBS proof.
- **Verifier:** Verifies the disclosed messages against the BBS proof with the signers **public** key

Assumes **no** special relationship between *signer* and *verifier*

Also corresponds to Verifiable Credential three party model

- **Signer → Issuer**
- **Prover → Holder**
- **Verifier → Verifier**

Blind BBS Signatures: Example Application

“In the BBS scheme, knowledge of a valid signature allows generation of BBS proofs. As a result, a signature compromise can lead to impersonation of the Prover by malicious actors. Using Blind BBS Signatures on the other hand, the Prover can commit to a secret message before issuance, guaranteeing that no one will be able to generate a valid proof without knowledge of their secret.”

In some applications this is known as “holder binding” or “key binding”

Anonymous Holder Binding

In other credential schemes adding “impersonation protection” via “key binding” ***unlinkability is typically lost,*** e.g., such schemes involve sharing a prover’s public key.

With the mechanisms of Blind BBS signatures unlinkability can be preserved, hence allowing “holder” binding to be anonymous. See the Anonymous Holder Binding feature in the W3C Data Integrity BBS Cryptosuites v1.0 CR.

Blind BBS Signatures: commitment and proof

Commitment (generalized Pedersen):

1. (Q_2, J_1, ..., J_M) = blind_generators
2. C = Q_2 * secret_prover_blind + J_1 * msg_1 + ... + J_M * msg_M

Where C is the binding and hiding *commitment* to the *provers* messages.

NIZKP of Commitment (Sigma protocol):

1. (secret_prover_blind, s~, m~_1, ..., m~_M) = get_random_scalars(M + 2)
2. 3. Cbar = Q_2 * s~ + J_1 * m~_1 + ... + J_M * m~_M
3. challenge = calculate_blind_challenge(C, Cbar, blind_generators, api_id)
4. s^ = s~ + secret_prover_blind * challenge
5. for m in (1, 2, ..., M): m^_i = m~_1 + msg_i * challenge
6. proof = (s^, (m^_1, ..., m^_M), challenge)

Blind BBS Signatures: API and Reuse

- **BlindSign**(SK, PK, commitment_with_proof, header, messages)
- **Verify**(PK, signature, header, messages, committed_messages, secret_prover_blind)
- **BlindProofGen**(PK, signature, header, ph, msgs, com_msgs, dis_indexes, dis_com_indexes, secret_prover_blind)
- **BlindProofVerify**(PK, proof, header, ph, L, dis_msgs, dis_com_msgs, dis_indexes, dis_com_indexes)

Reuse from BBS Signature Scheme:

- BBS.CoreVerify
- BBS.CoreProofGen
- BBS.create_generators
- BBS.messages_to_scalars
- BBS.get_random_scalars
- BBS.hash_to_scalar
- BBS.calculate_domain

BBS Per Verifier Linkability (Pseudonyms)

“Each BBS proof reveals no information other than the signed messages that the **Prover** chooses to disclose in that specific instance. As such, the **Verifier** of the BBS proof, may not be able to track those presentations over time. Although in many applications this is desirable, there are use cases that require the **Verifier** be able to ***track*** the BBS proofs they receive from the same Prover.”

Limited linkability, or limited tracking, or remember me but not too much…

“A ***pseudonym***, is a value that will remain constant each time a **Prover** presents a BBS proof to the same **Verifier**, but will be different (and unlinkable), when the **Prover** interacts with a different **Verifier**. This provides a way for a recipient (**Verifier**) to track the presentations intended for them, while also *hindering* them from *tracking* the **Prover's** interactions with other **Verifiers**.”

BBS Per Verifier Linkability: Pseudonyms

Computation:

Pseudonym = hash_to_curve_g1(*context_id*) * *nym_secret*

Calculated by prover at proof generation time

nym_secret only known to **prover** bound to signature generated by **signer**

“The **Context Identifier** (context_id) is an octet string that represents a specific context of usage, within which, the ***pseudonym*** will have a constant value. Context Identifiers can take the form of unique Verifier Identifiers, Session Identifiers etc., depending on the needs of the application. **Verifiers** will be able to use the Pseudonym values to track the presentations generated by a **Prover**, using the same signature, for that specific context.”

Not just per verifier! See Pseudonyms with Hidden PID feature in W3C Data Integrity BBS Cryptosuites v1.0.

BBS Per Verifier Linkability: API and Reuse

API is under development! Please provide feedback!

- CommitmentWithNym(...)
- BlindSignWithNym(...)
- BindVerifyWithNym(...)
- ProofGenWithNym(...)
- ProofVerifyWithNym(...)

Reuse from BBS Signature Scheme and Blind BBS Signatures:

- BBS.create_generators
- BBS.messages_to_scalars
- Blind.deserialize_and_validate_commit
- Blind.FinalizeBlindSign
- BBS.get_random_scalars
- BBS.hash_to_scalar
- BBS.calculate_domain
- BBS.ProofInit
- BBS.ProofFinalize
- BBS.ProofVerifyInit

Additional Information

Presentations

- BBS+ Applications, Standardization, and a Bit of Theory , NIST Crypto Reading Club Presentation (virtual) - October 2023, Greg Bernstein and Vasilis Kalos.
- BBS for Verifiable Credentials - Data Integrity, June 4th, 2024.
- BBS Advanced Features, June 25th, 2024.
- BBS Advanced Features 2, September 3rd, 2024.
- In browser JavaScript only BBS Demo

Papers and References

- A Graduate Course in Applied Cryptography (B&S), Dan Boneh and Victor Shoup, version 0.6 (latest version, Jan. 2023).
- Paper 2023/275 Revisiting BBS Signatures, Stefano Tessaro and Chenzhi Zhu, also EUROCRYPT 2023. Optimizations and additional security proofs.
- Paper 2016/663 Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited , Jan Camenisch, Manu Drijvers, and Anja Lehmann. Sections 4.3-4.5 are basis for current IETF draft.