Key Blinding for Signature Schemes

draft-dew-cfrg-signature-key-blinding
Setting: Single Prover

Unforgeability: Given (msg, pk, σ), will the Verifier conclude that the owner of sk produced σ with overwhelming probability? ✅
Setting: Single Prover

Unforgeability: Given $(msg, pk, \sigma)$, will the Verifier conclude that the owner of $sk$ produced $\sigma$ with overwhelming probability?

... but what if one wanted the signature or public key to reveal nothing about the Prover?

- Tor hidden service identity blinding protocol: Signing hidden service descriptor
- Privacy Pass rate limiting: Signing Privacy Pass token requests
- Cryptocurrency private airdrop: Computing public airdrop tokens
- BIP32 wallets
Setting: Multiple Provers

Prover 0
\[ \sigma_0 = \text{Sign}(sk_0, \text{msg}) \]

Mediator
\[ \begin{align*}
  a &= \text{Verify}(pk_b, \text{msg}, \sigma_b) \\
  b' &= ?
\end{align*} \]

Prover 1
\[ \sigma_1 = \text{Sign}(sk_1, \text{msg}) \]

Verifier
\[ (pk_0', pk_1) \]

\[ b \leftarrow \{0,1\}^* \]
Setting: Multiple Provers

Prover 0
\( \sigma_0 = \text{Sign}(sk_0, \text{msg}) \)

Prover 1
\( \sigma_1 = \text{Sign}(sk_1, \text{msg}) \)

Mediator
\( b \leftarrow \{0,1\}^* \)

Verifier
\( a = \text{Verify}(pk_b, \text{msg}, \sigma_b) \)

\( b' = ? \)

1. Unforgeability: Given \((\text{msg}, \text{pk}_b, \sigma_b)\), will the Verifier conclude that the owner of \(sk_b\) produced \(\sigma_b\) with overwhelming probability? ✔

2. Unlinkability: Given \((\text{msg}, \text{pk}_b, \sigma_b)\), can the Verifier determine \(b\) with probability not negligibly better than 1/2?
1. **Unforgeability**: Given \((\text{msg}, \text{pk}_b, \sigma_b)\), will the Verifier conclude that the owner of \(\text{sk}_b\) produced \(\sigma_b\) with overwhelming probability?

2. **Unlinkability**: Given \((\text{msg}, \text{pk}_b, \sigma_b)\), can the Verifier determine \(b\) with probability not negligibly better than 1/2? ❌
Functional Requirements

Unforgeable signature scheme with the following additional properties:

1. Per-message public keys are independently distributed from long-term public keys
2. Per-message signatures do not leak any information about the long-term signing keys

Proposed solution: signature schemes with key blinding
Signature Scheme with Key Blinding

Extend digital signature schemes with two functionalities

1. BlindPublicKey and UnblindPublicKey: Given public key and blinding key, produce blinded public key that is independent from input public key

   \[
   \text{UnblindPublicKey(BlindPublicKey}(\text{pk}_S, \text{bk})), \text{bk}) = \text{pk}_S
   \]

2. BlindKeySign: Sign message with signing key and blinding key, producing a signature that is independent of the input signing key.

   \[
   \text{Verify(BlindPublicKey}(\text{pk}_S, \text{bk}), \text{msg}, \text{BlindKeySign}(\text{sk}_S, \text{bk}, \text{msg})) = 1
   \]
Setting: Multiple Provers

Prover 0

\( \text{bk}_0 \leftarrow \text{KeyGen()} \)
\( \sigma_0 = \text{BlindKeySign}(\text{sk}_0, \text{bk}_0, \text{msg}) \)

Prover 1

\( \text{bk}_1 \leftarrow \text{KeyGen()} \)
\( \sigma_1 = \text{BlindKeySign}(\text{sk}_1, \text{bk}_1, \text{msg}) \)

Mediator

\( \sigma_b \) is the output of a secure BlindKeySign algorithm

\( b \leftarrow \{0,1\}^* \)
\( \text{bk} = \text{BlindPublicKey}(\text{pk}_b, \text{bk}_b) \)

Verifier

\( a = \text{Verify}(\text{bk}, \text{msg}, \sigma_b) \)
\( b' = ? \)

1. Unforgeability: Given \((\text{msg}, \text{bk}, \sigma_b)\), will the Verifier conclude that the owner of \(\text{sk}_b\) produced \(\sigma_b\) with overwhelming probability? ✓

2. Unlinkability: Given \((\text{msg}, \text{bk}, \sigma_b)\), can the Verifier determine \(b\) with probability not negligibly better than 1/2?
1. Unforgeability: Given \((\text{msg}, \text{bk}, \sigma_b)\), will the Verifier conclude that the owner of \(sk_b\) produced \(\sigma_b\) with overwhelming probability?

2. Unlinkability: Given \((\text{msg}, \text{bk}, \sigma_b)\), can the Verifier determine \(b\) with probability not negligibly better than 1/2? ✔️
Current Status

Implementation status:

• PureEdDSA (RFC8032) and ECDSA key blinding extension support and test vectors

• Several interoperable implementations exist

Unlinkability and unforgeability analysis for EdDSA and ECDSA variants is underway
1) Interest in working on signature schemes with key blinding support?

2) Interest in adopting this document?