Security Analysis of Signature Schemes with Key Blinding

https://ia.cr/2023/380

Edward Eaton, Tancrède Lepoint, Christopher A. Wood

March 31, 2023

Summary

- Security proofs for variations of signature schemes as described in `draft-irtf-cfrg-signature-key-blinding`
- Schemes based on ECDSA and EdDSA
- Discussions of security models
- Consider configurations of schemes and their impact on security

Full paper available now: `https://ia.cr/2023/380`

What is key-blinding?

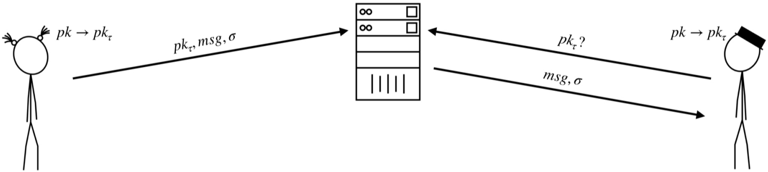
Consider an EC signature key pair $(sk, pk) = (a, a \cdot G) = (a, A)$.
Take another scalar b and compute $B = b \cdot A$. Then:

- With just B, you don't know anything about A
- You can sign messages under ab
- Just B is enough to verify signatures

'Decoupled' A and B except original sk is still needed to sign!

What is key-blinding?

Useful when signature schemes are used in anonymity networks.



What is key-blinding?

- $\mathsf{KeyGen}() \rightarrow (pk, sk)$ *(generate the 'identity' keypair)*
- $\mathsf{BlindPK}(pk, bk) \rightarrow bpk$ *(blind the ID public key with respect to bk)*
- $\mathsf{Sign}(msg, sk, bk) \rightarrow \sigma$ *(sign a message with respect to bk)*
- $\mathsf{Vrfy}(msg, \sigma, bpk) \rightarrow 0/1$ *(verify with respect to a blinded public key)*
- OPTIONAL: $\mathsf{UnblindPK}(bpk, bk) \rightarrow pk$ *(undo the blinding operation)*

Correctness: $\mathsf{Vrfy}(msg, \mathsf{Sign}(msg, sk, bk), \mathsf{BlindPK}(pk, bk)) = 1$

Security Models

Unforgeability:

- Give adversary identity public key pk
- Adversary makes queries $\mathsf{Sign}(msg, bk)$, gets back σ
- Adversary submits (msg^*, σ^*, bk^*)

Adversary wins if $\mathsf{Vrfy}(msg^*, \sigma^*, \mathsf{BlindPK}(pk, bk^*)) = 1$ and freshness condition is met:

1. msg^* was never part of a signing query *(basic unforgeability)*
2. msg^*, bk^* was never a signing query *(bk-binding unforgeability)*
3. σ^* was not the result of a query (msg^*, bk^*) *(bk-binding strong unforgeability)*

Security Models

Unlinkability:

- Adversary queries BlindPK() gets back $bpk = \mathrm{BlindPK}(pk, bk)$ for random bk
- Adversary can query $\mathrm{Sign}(msg, bpk)$ for any bpk previously returned from BlindPK
- Adversary submits a challenge query, gets back either $\mathrm{BlindPK}(pk, bk)$ or $\mathrm{BlindPK}(pk', bk)$ for fresh pk' and random bk.

Adversary wins if they guess whether challenge blinded public key used identity key or not.

If scheme doesn't support UnblindPK: Permit bk to be adversarially controlled.

Constructions for key-blinding

<u>ECDSA</u>.BlindPK(pk, bk)

- $bk \in \{0,1\}^{256}$
- $\beta \leftarrow H2S(bk)$, $\beta \in \mathbb{Z}_q^*$
- Return $bpk = \beta \cdot pk$

<u>Ed25519</u>.BlindPK(pk, bk)

- $bk \in \{0,1\}^{256}$
- $h \leftarrow SHA512(bk)$
- $\beta \leftarrow h[0:32]$
- Return $bpk = \beta \cdot pk$

Signing must be appropriately modified to match what the induced "blinded secret key" is. As well, when calculating $k \leftarrow SHA512(R, A', msg)$ in Ed25519 signing, $A' = bpk$.

Our Results

- A proof of unlinkability for Ed25519 (seperated from Tor context)
- A tight proof of the bk-binding strong unforgeability of Ed25519
- A proof of unlinkability for ECDSA
- A proof of plain unforgeability in the ECGGM for ECDSA
- Benchmarks for Ed25519 and ECDSA (P-384 and SHA-384)

`https://ia.cr/2023/380`