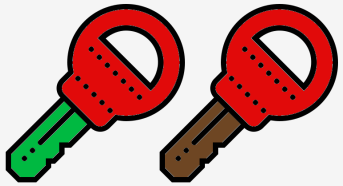
SoK: Signatures With Randomizable Keys

Sofía Celi, Scott Griffy, Lucjan Hanzlik,
Octavio Perez Kempner, Daniel Slamanig

Brave Software, Brown University, CISPA,
NTT Social Informatics Laboratories, AIT Austrian Institute of Technology

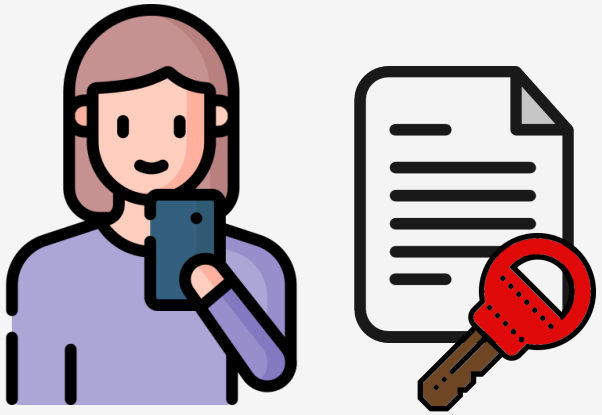
Digital Signatures

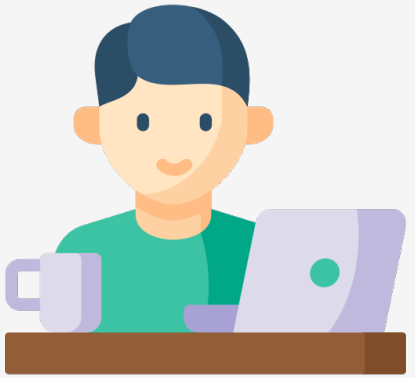
Digital Signatures



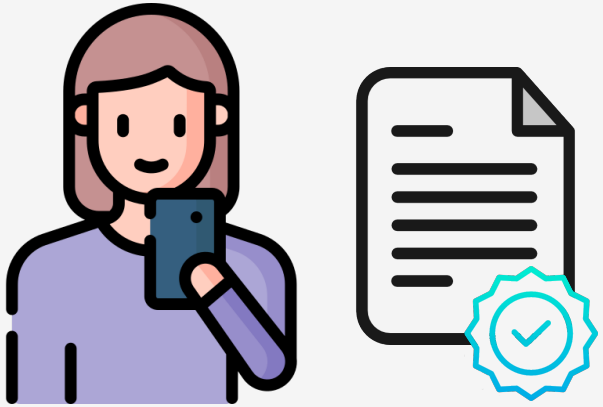
Digital Signatures

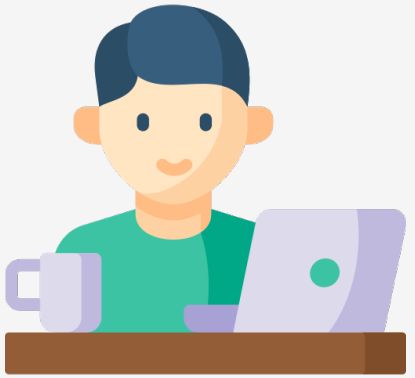
Digital Signatures



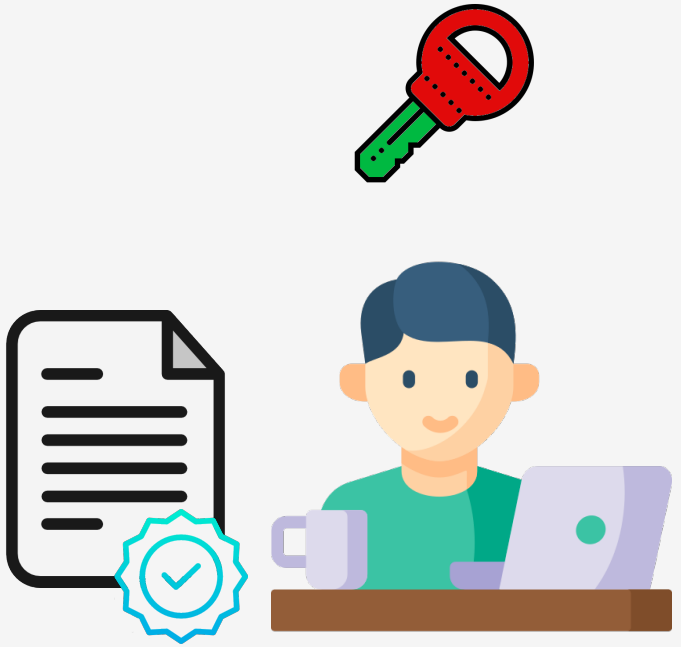


Digital Signatures



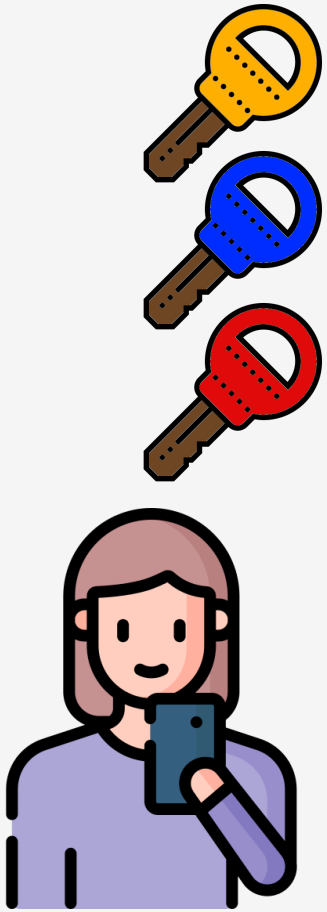


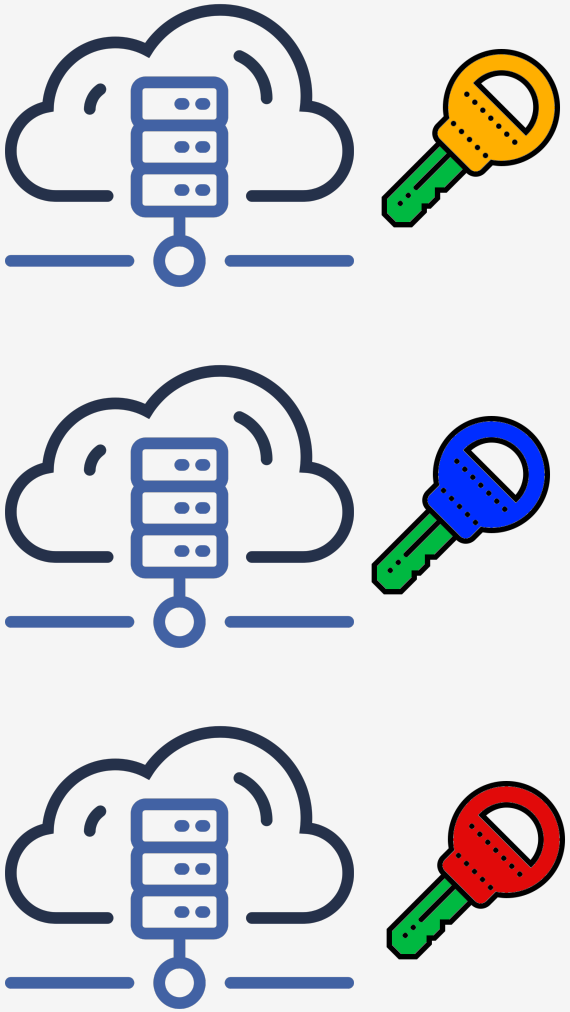
Digital Signatures



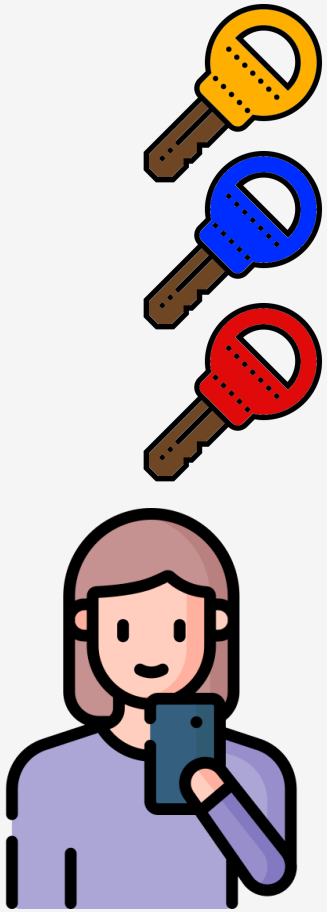
WebAuthn Application

WebAuthn Application

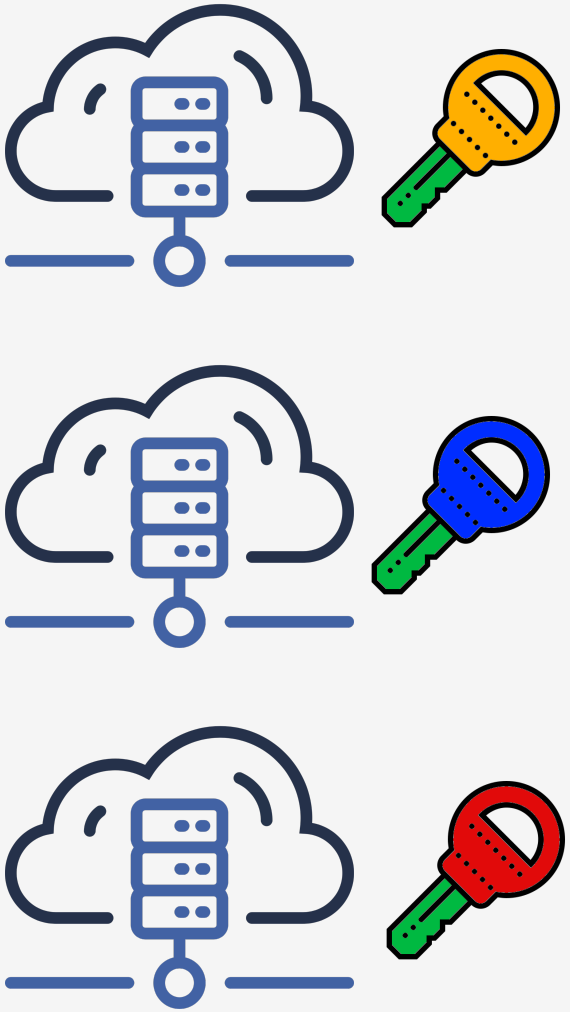




WebAuthn Application

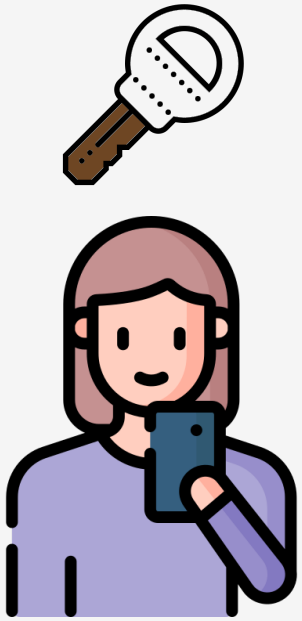


Can we do better?



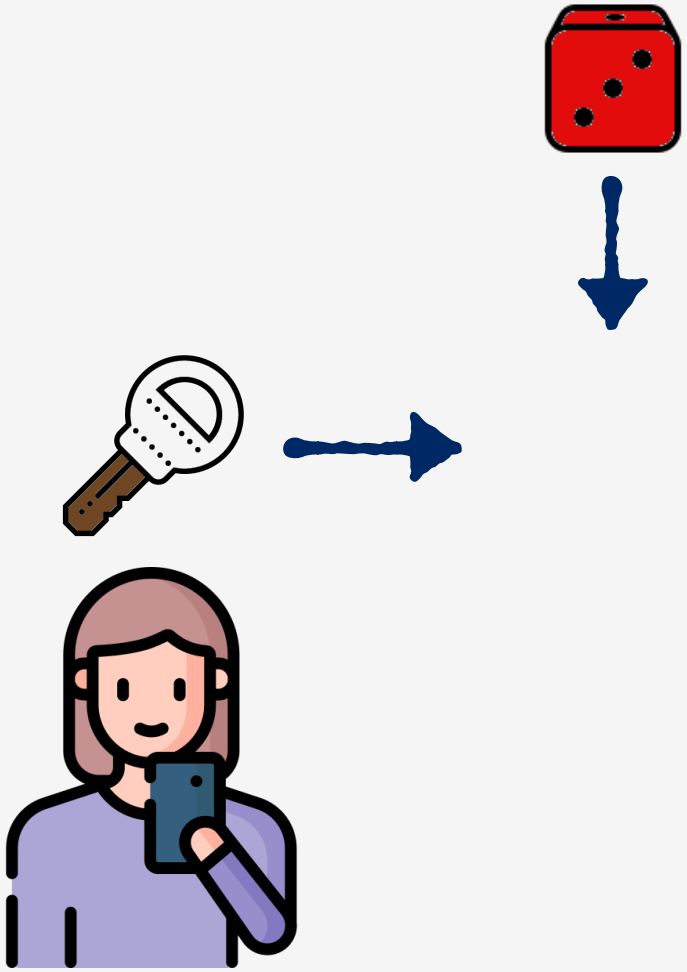
Digital Signatures: Single Master Key

Digital Signatures: Single Master Key

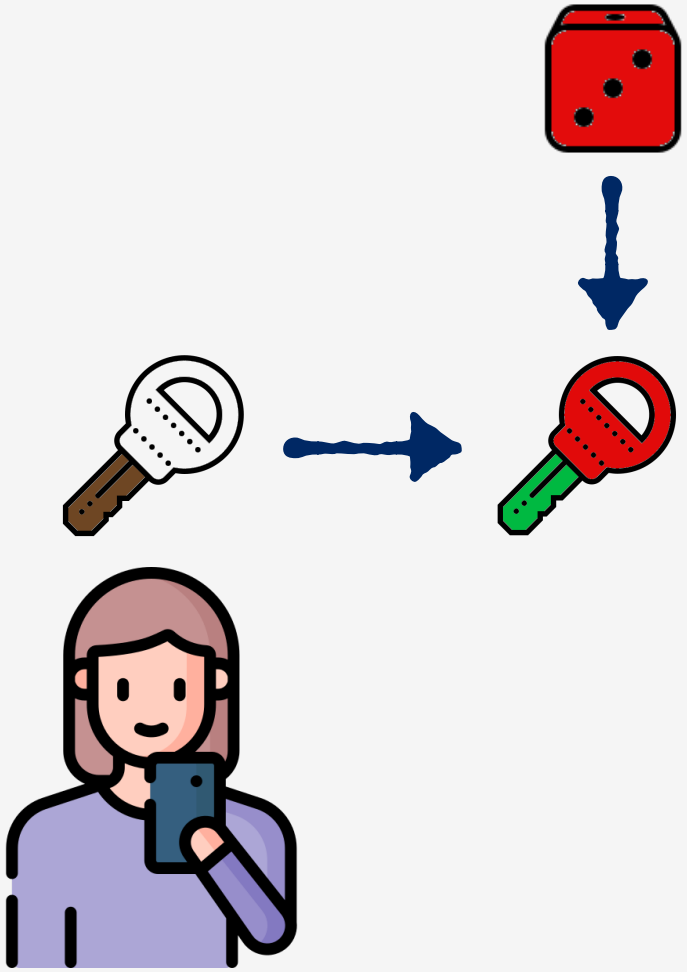


Digital Signatures: Single Master Key

Digital Signatures: Single Master Key



Digital Signatures: Single Master Key

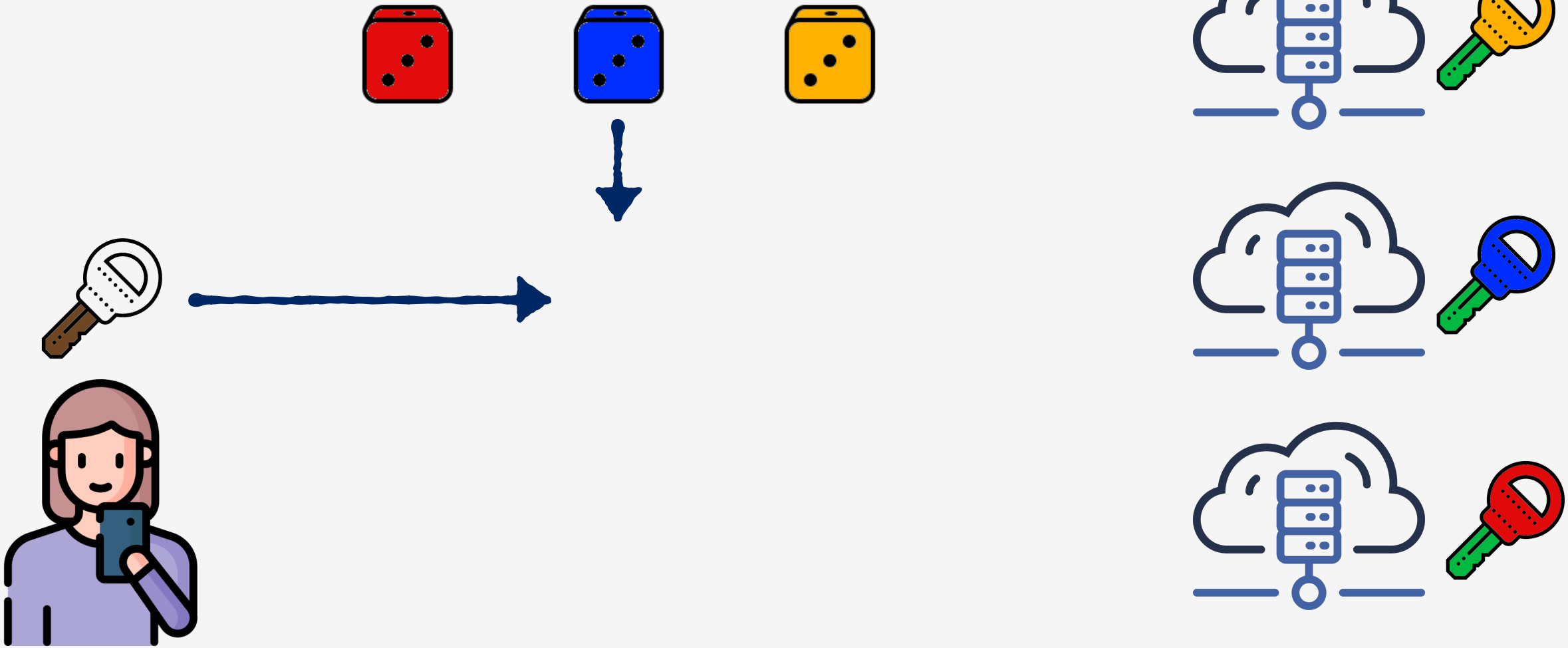


Digital Signatures: Single Master Key

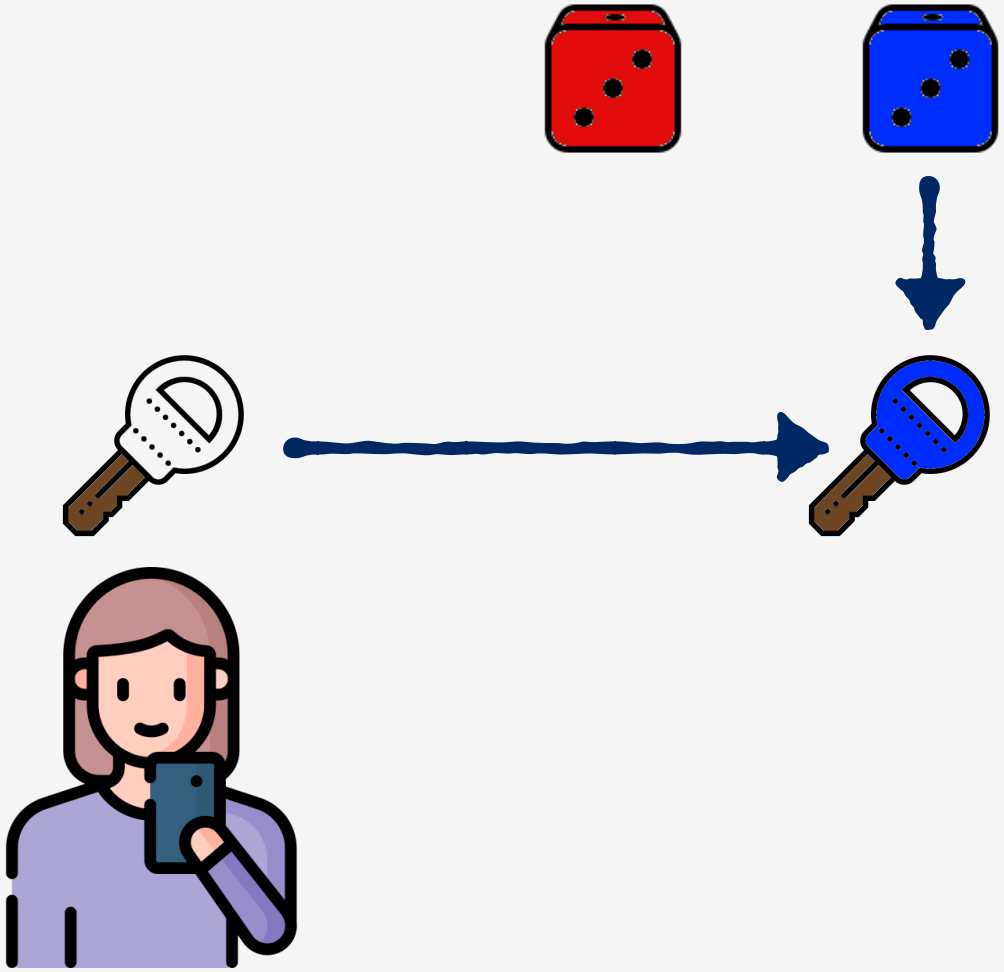
Digital Signatures: Single Master Key

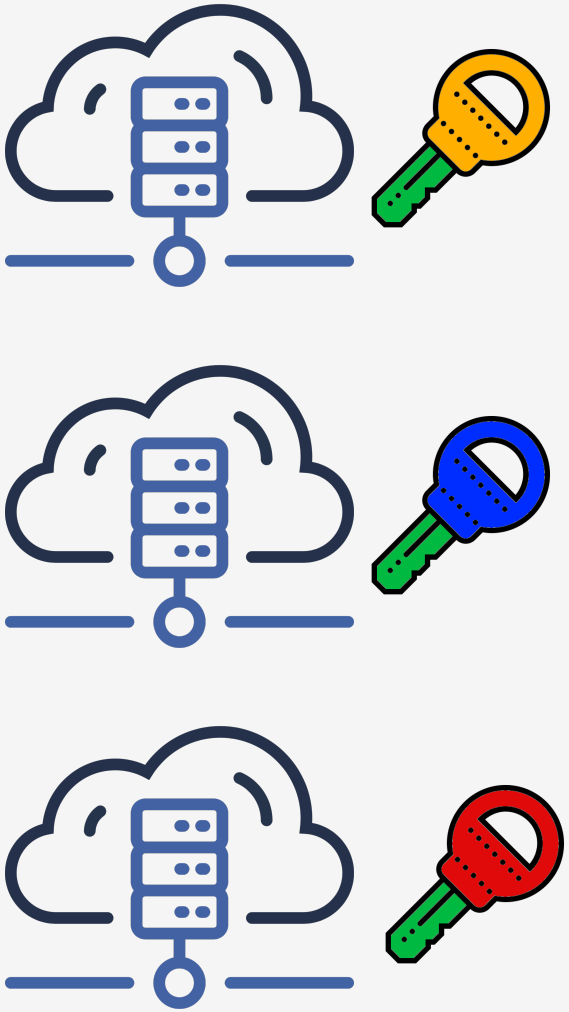
Digital Signatures: Single Master Key

Digital Signatures: Single Master Key



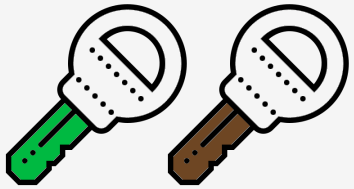
Digital Signatures: Single Master Key



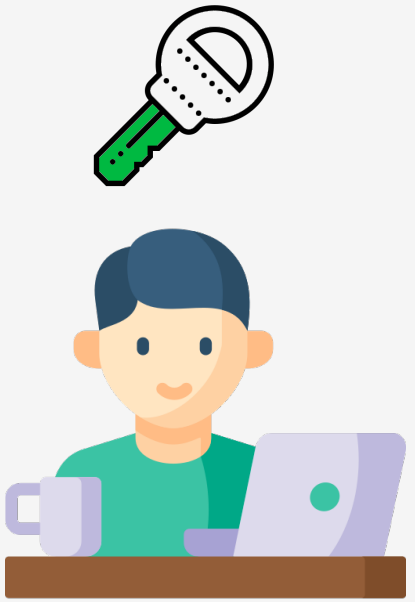


Digital Signatures: Single Master Key

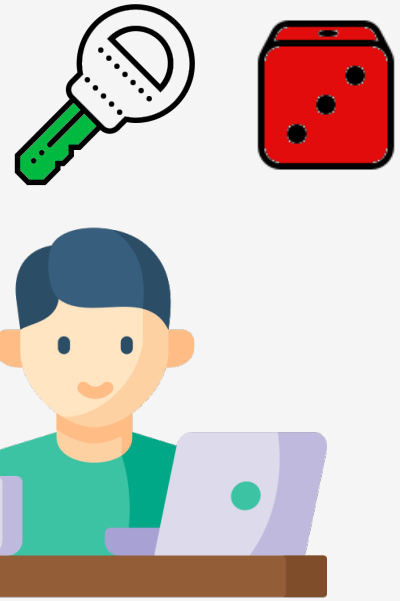
Digital Signatures: Public Randomization



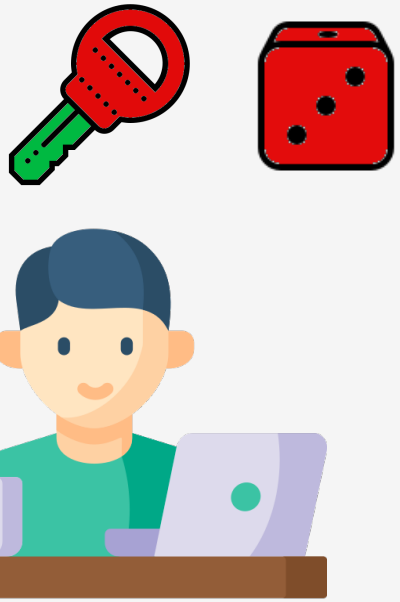
Digital Signatures: Public Randomization



Digital Signatures: Public Randomization

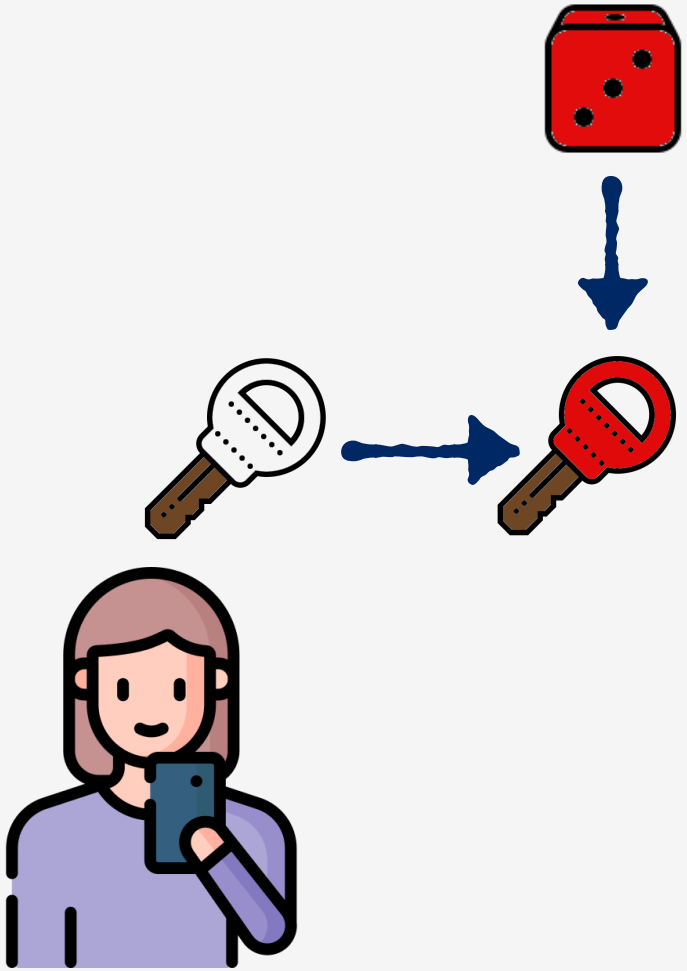


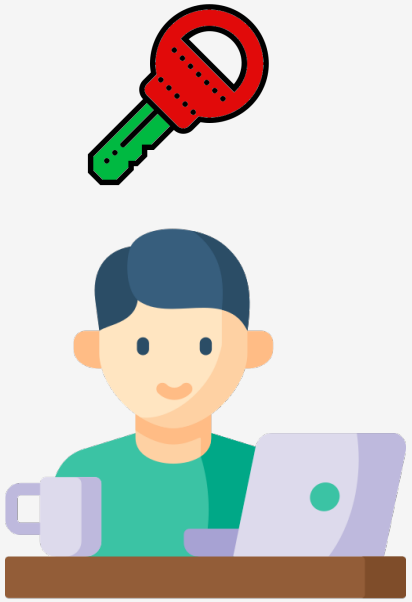
Digital Signatures: Public Randomization



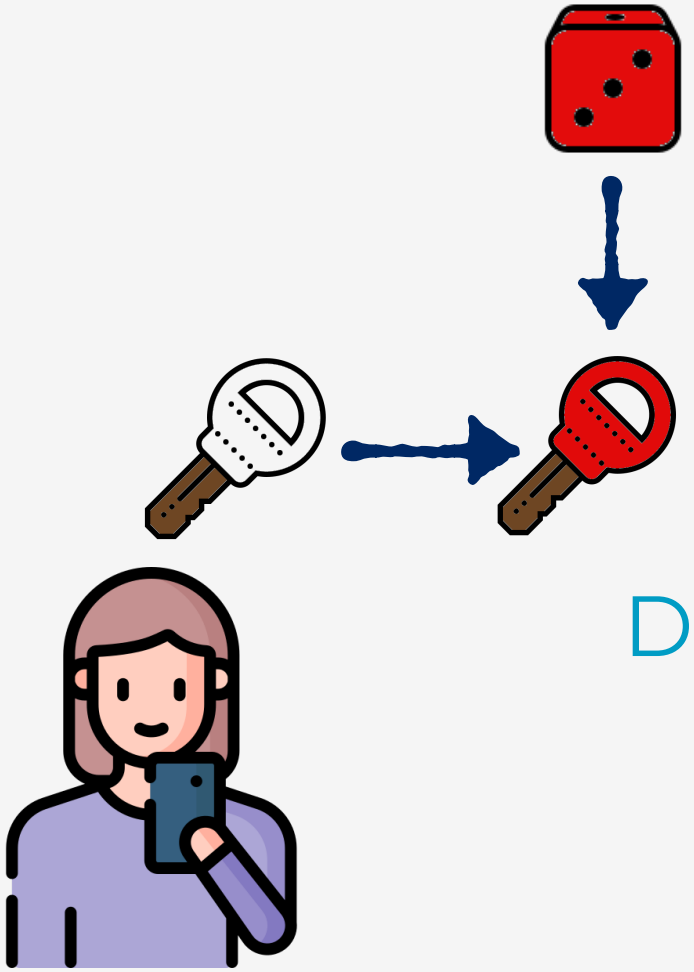
Digital Signatures: Public Randomization

Digital Signatures: Public Randomization





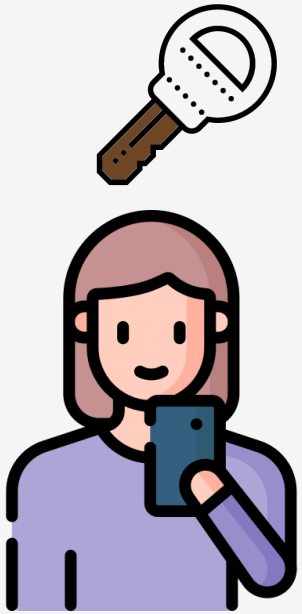
Digital Signatures: Public Randomization



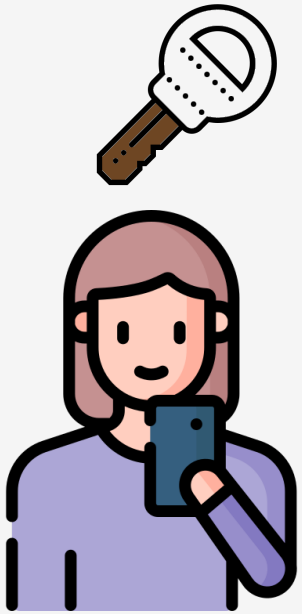
Deterministic Wallets

Stealth Addresses

Digital Signatures: Randomizing Signature



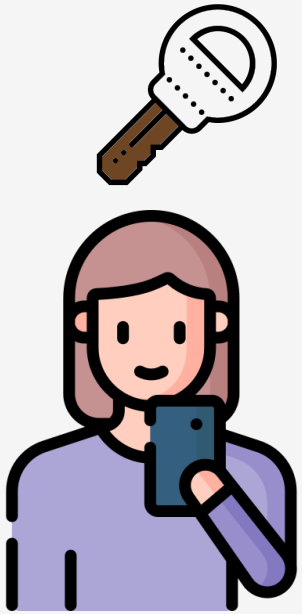
Digital Signatures: Randomizing Signature



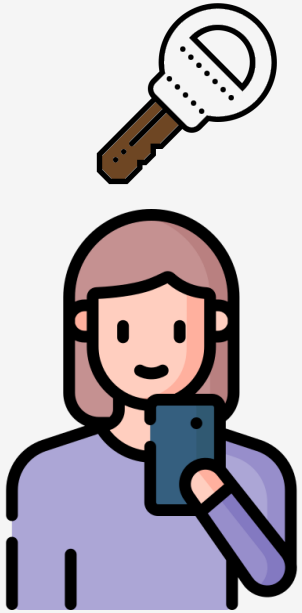
Digital Signatures: Randomizing Signature

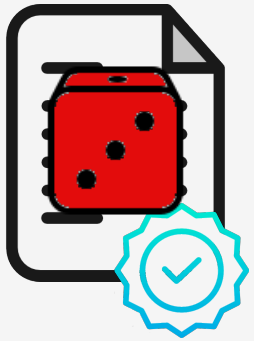
Digital Signatures: Randomizing Signature

Digital Signatures: Randomizing Signature

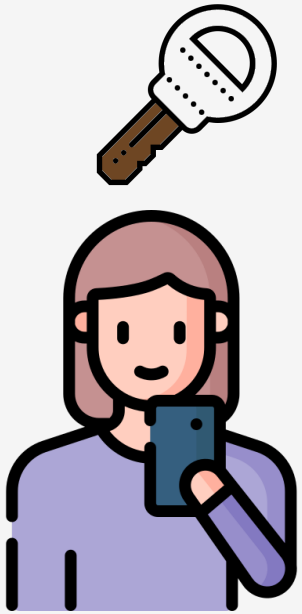


Digital Signatures: Randomizing Signature

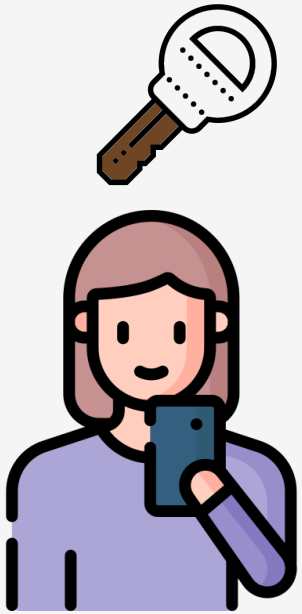




Digital Signatures: Randomizing Signature



Digital Signatures: Randomizing Signature



Digital Signatures: Different Notions

Digital Signatures: Different Notions

* Sok (in Polish) = Juice

Our Contribution

- Introduce signatures with randomizable keys
  - extend digital signatures
  - parametrizable security properties

- Revisit prior work and how it relates to our syntax and model

- Show what is required for specific applications

New Algorithms: Randomization and Adaptation

New Algorithms: Randomization and Adaptation

- Separate algorithms to randomize secret/public key (RandSK & RandPK)
  - take as input the original key and key randomizer
  - outputs randomized key, e.g., pk’ = T(pk, r)

New Algorithms: Randomization and Adaptation

- Separate algorithms to randomize secret/public key (RandSK & RandPK)
  - take as input the original key and key randomizer
  - outputs randomized key, e.g., pk’ = T(pk, r)

- Optional adaptation algorithm
  - takes as input signature, public key and key randomizer
  - outputs signature valid under pk’
  - adapted signatures look like fresh signatures - (perfect) adaptation

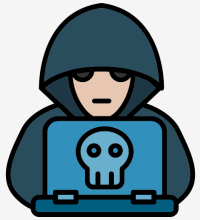
New Security Properties

- Unforgeability - no forged signatures
- Unlinkability - randomized public keys are not linkable to original ones
- Unextractability - cannot go back to original public key even knowing the key randomizer

Security Properties: α-Unforgeability

(sk,pk)

pk



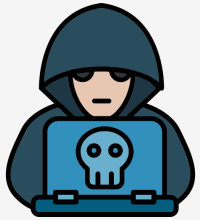
Security Properties: α-Unforgeability

(sk,pk)

get key randomizer

r

pk



Security Properties: α-Unforgeability

(sk,pk)

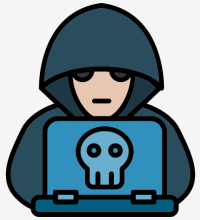
get key randomizer

r

m, r

σ valid under pk' = T(pk,r)

pk



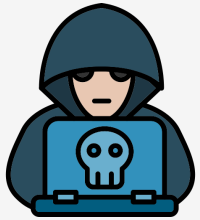
Security Properties: α-Unforgeability

(sk,pk)

get key randomizer

r

pk



m, r

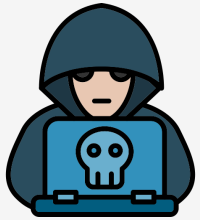
σ valid under pk' = T(pk,r)

from Oracle (if α=0)

Security Properties: α-Unforgeability

(sk,pk)

pk



get key randomizer

r

m, r

σ valid under pk' = T(pk,r)

from Oracle (if α=0)

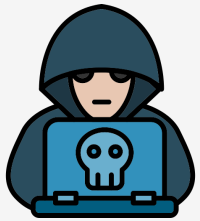
.
.
.

m*, σ*, r*

Security Properties: α-Unforgeability

(sk,pk)

pk



get key randomizer

r

m, r

σ valid under pk' = T(pk,r)

from Oracle (if α=0)

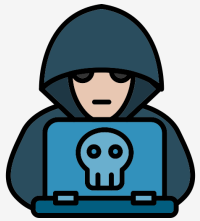
.
.
.

m*, σ*, r*

Adv wins iff

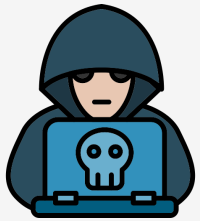
- σ* valid for pk* = T(pk,r*)
- r* from Oracle (if α=0)

Security Properties: (α,β,γ)-Unlinkability



Security Properties: (α,β,γ)-Unlinkability

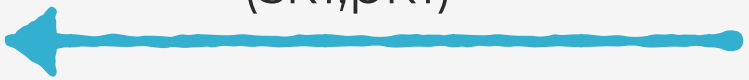
(sk0,pk0) β=0
(sk1,pk1)

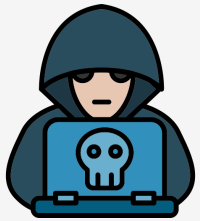


Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0) β=0
(sk1,pk1)

(sk0,pk0) β=1
(sk1,pk1)



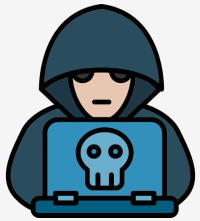


Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0) β=0
(sk1,pk1)

picks bit b and r

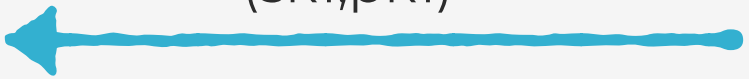
(sk0,pk0) β=1
(sk1,pk1)

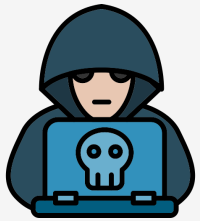


Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0) β=0
(sk1,pk1)

(sk0,pk0) β=1
(sk1,pk1)





picks bit b and r

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0) β=0
(sk1,pk1)

(sk0,pk0) β=1
(sk1,pk1)

picks bit b and r

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

(pk0, pk1, pk*)

Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0) β=0
(sk1,pk1)

(sk0,pk0) β=1
(sk1,pk1)

picks bit b and r

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

(pk0, pk1, pk*)

signing oracle

Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0) β=0
(sk1,pk1)

(sk0,pk0) β=1
(sk1,pk1)

picks bit b and r

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

(pk0, pk1, pk*)

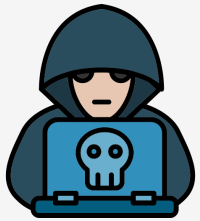
signing oracle

b*

Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0) β=0
(sk1,pk1)

(sk0,pk0) β=1
(sk1,pk1)



picks bit b and r

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

(pk0, pk1, pk*)

signing oracle

b*

Adv wins iff b = b*

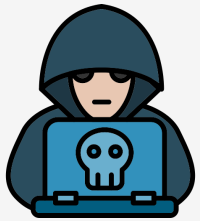
Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0)
(sk1,pk1)

β=0

(sk0,pk0)
(sk1,pk1)

β=1



picks bit b and r

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

(pk0, pk1, pk*)

signing oracle

no queries (if γ=0)
for key pk* (if γ=1)
for all keys (if γ=3)

b*

Adv wins iff b = b*

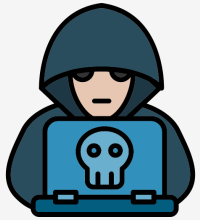
Security Properties: (α,β,γ)-Unlinkability

(sk0,pk0)
(sk1,pk1)

β=0

(sk0,pk0)
(sk1,pk1)

β=1



picks bit b and r

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

(pk0, pk1, pk*)

signing oracle

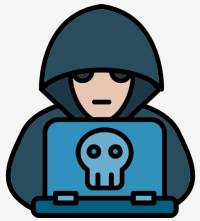
no queries (if γ=0)
for key pk* (if γ=1)
for all keys (if γ=3)

b*

Adv wins iff b = b*

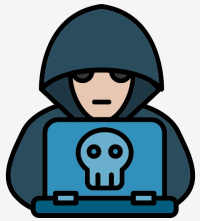
Public parameters generated honestly (α=0) or by adversary (α=1)

Security Properties: (α,β)-Unextractability



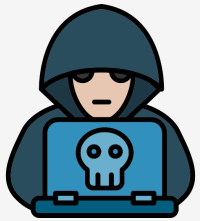
Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)



Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

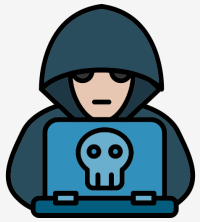


picks bit b and r (if β=0)

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1

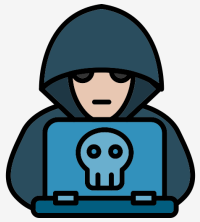


picks bit b and r (if β=0)

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1



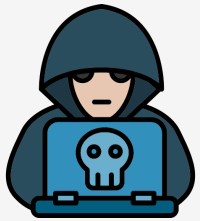
picks bit b and r (if β=0)

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1



picks bit b and r (if β=0)

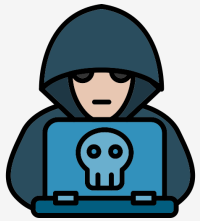
sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

pk*, r

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1



picks bit b and r (if β=0)

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

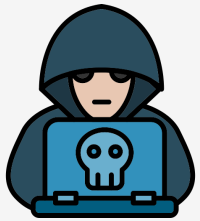
pk*, r

signing oracle

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1



picks bit b and r (if β=0)

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

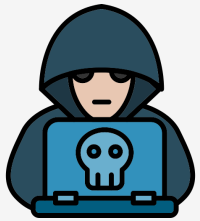
pk*, r

signing oracle access to all keys

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1



picks bit b and r (if β=0)

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

pk*, r

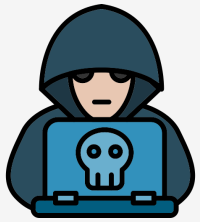
signing oracle access to all keys

b*

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1



picks bit b and r (if β=0)

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

pk*, r

signing oracle access to all keys

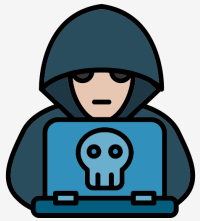
b*

Adv wins iff b = b*

Security Properties: (α,β)-Unextractability

(sk0,pk0)
(sk1,pk1)

r β=1



picks bit b and r (if β=0)

sk* = RandSK(skb,r)
pk* = RandPK(pkb,r)

pk*, r

signing oracle access to all keys

b*

Adv wins iff b = b*

Public parameters generated honestly (α=0) or by adversary (α=1)

BLS Signatures - Next Talk

- Support perfect adaptation
- Unforgeable against malicious randomizer (1-UNF)
- (1,1,3)-Unlinkable but are not unextractable
- Can be used for Deterministic Wallets and Stealth Addresses which can work with (0,0,3)-UNL and 1-UNF

BLS Signatures - Next Talk

- Support perfect adaptation
- Unforgeable against malicious randomizer (1-UNF)
- (1,1,3)-Unlinkable but are not unextractable
- Can be used for Deterministic Wallets and Stealth Addresses which can work with (0,0,3)-UNL and 1-UNF

See paper for full systematization

Thank you for your attention