RSACONFERENCE2014

FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Share.
Learn.
Secure.

Capitalizing on
Collective Intelligence

Rethinking Verifiably Encrypted Signatures: A Gap in Functionality and Potential Solutions

SESSION ID: cryp-r02

Sarah Meiklejohn

Graduate researcher
UC San Diego

Models for cryptographic primitives

Models for cryptographic primitives

- In cryptography, we put a lot of effort into accurately modeling **security**: what an adversary can and can't do

Models for cryptographic primitives

- In cryptography, we put a lot of effort into accurately modeling **security**: what an adversary can and can't do

- But it's also incredibly important to accurately model **functionality**!

Models for cryptographic primitives

- In cryptography, we put a lot of effort into accurately modeling **security**: what an adversary can and can't do

- But it's also incredibly important to accurately model **functionality**!

- We look at definitions for verifiably encrypted signatures (VES)

Models for cryptographic primitives

- In cryptography, we put a lot of effort into accurately modeling **security**: what an adversary can and can't do
- But it's also incredibly important to accurately model **functionality**!
- We look at definitions for verifiably encrypted signatures (VES)
  - First show a generic construction based **solely on signatures**

Models for cryptographic primitives

- In cryptography, we put a lot of effort into accurately modeling **security**: what an adversary can and can't do
- But it's also incredibly important to accurately model **functionality**!
- We look at definitions for verifiably encrypted signatures (VES)
  - First show a generic construction based **solely on signatures**
  - Then propose **new definition(s)**

An application: fair contract signing [ASW88]

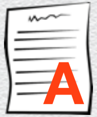
- Alice and Bob want each other's signature on a particular document, but they don't trust each other

An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other

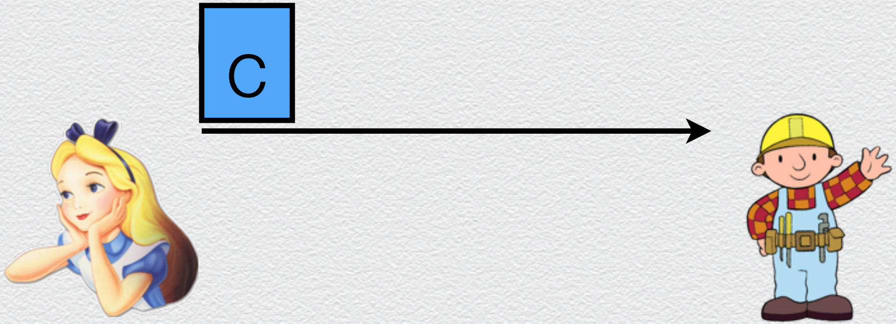
An application: fair contract signing [ASW88]

- Alice and Bob want each other’s signature on a particular document, but they don’t trust each other



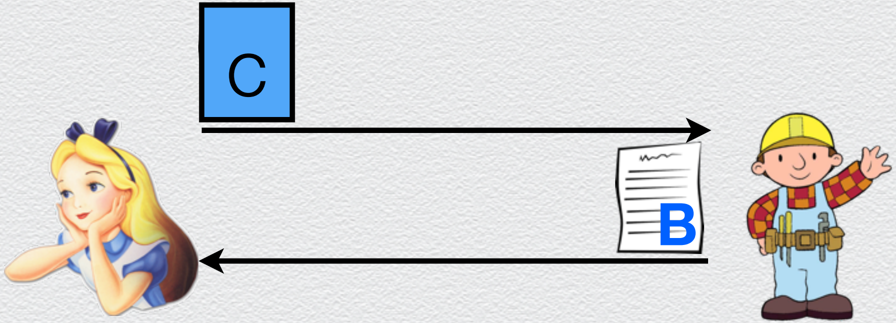
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



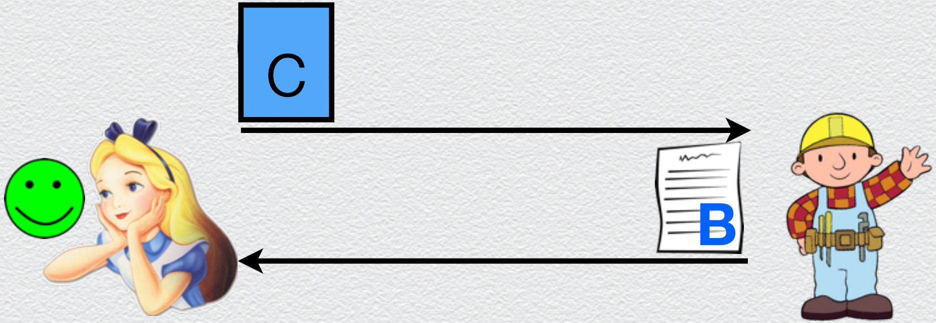
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



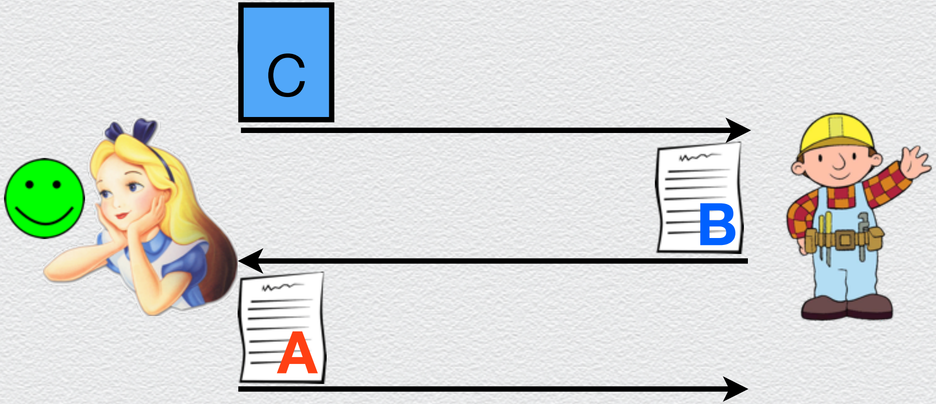
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



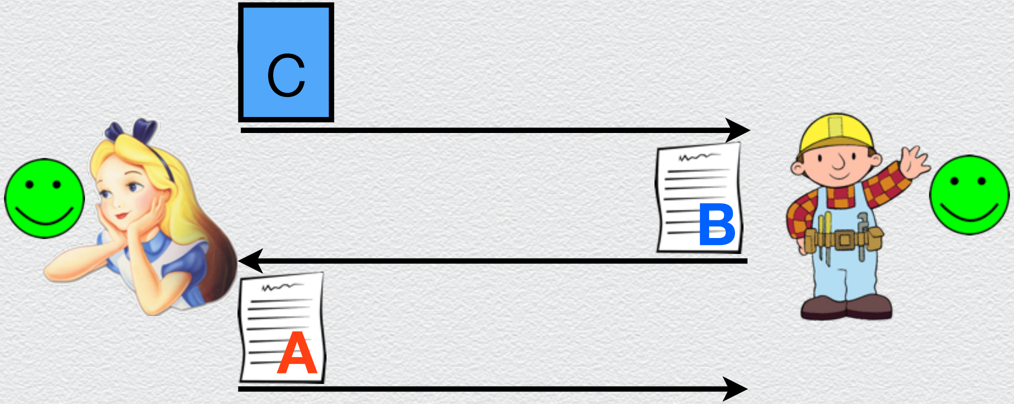
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



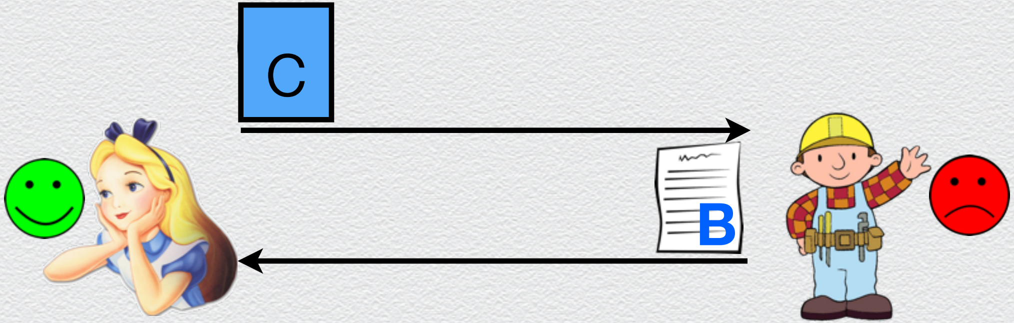
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



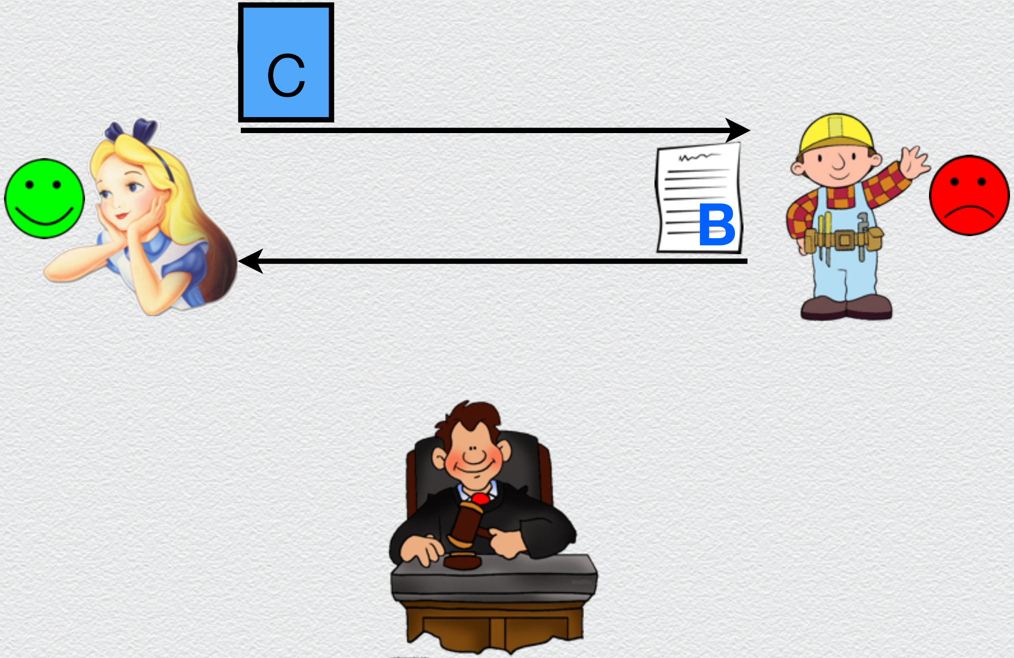
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



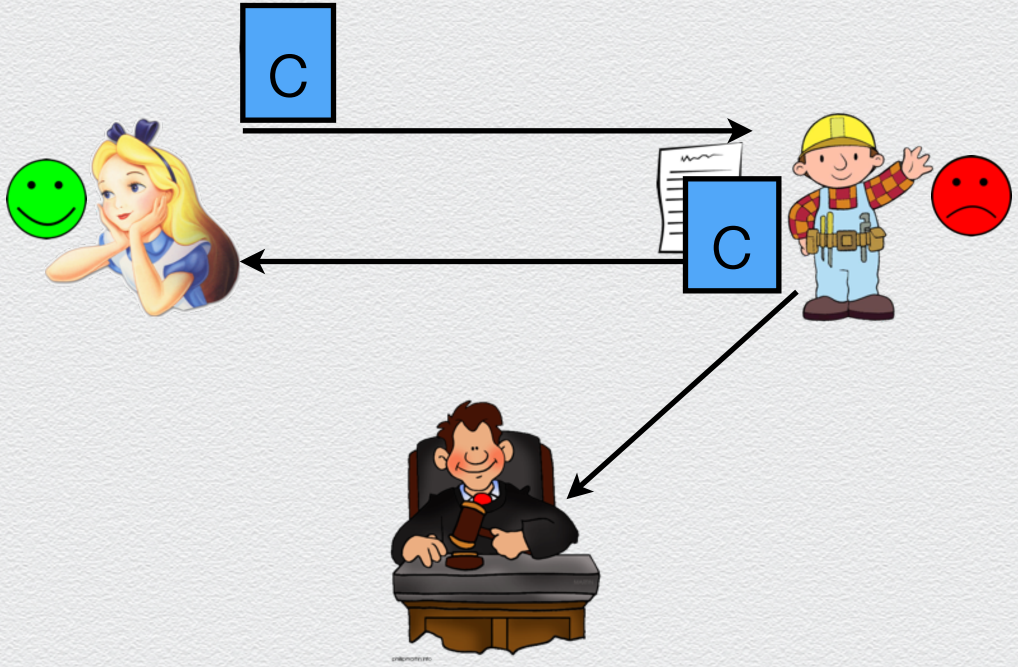
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



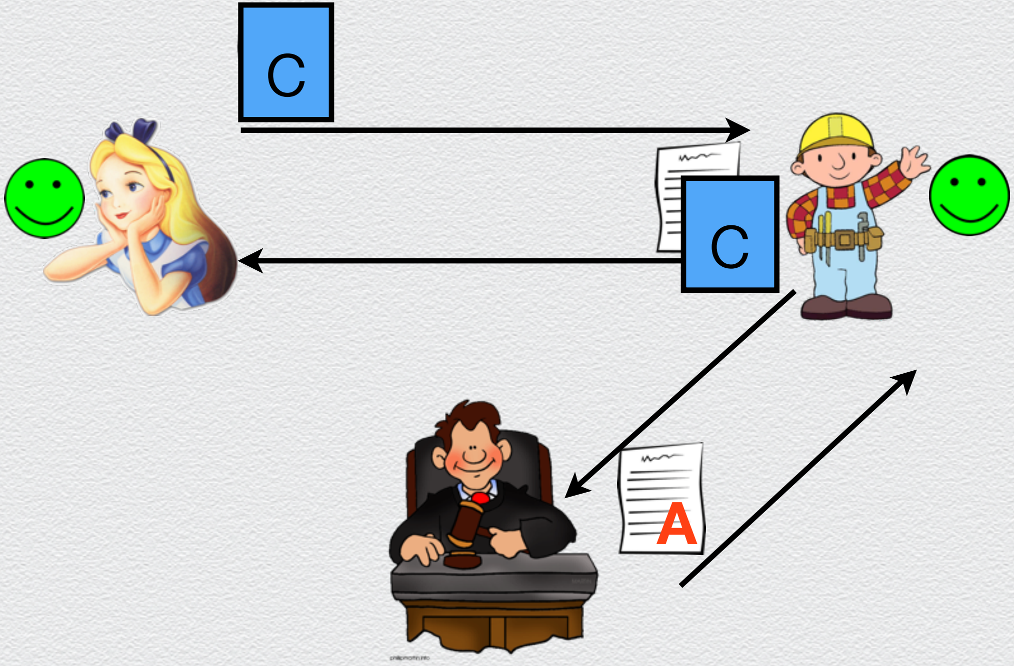
An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



An application: fair contract signing [ASW88]

- Alice and Bob want each other's signature on a particular document, but they don't trust each other



Definitions for VES:

- Unforgeability
- Opacity
- Extractability

Verifiably encrypted signatures [BGLS03]

Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

$(pk_A, sk_A) \leftarrow KG(1^k)$

$(pk_B, sk_B) \leftarrow KG(1^k)$

$(apk, ask) \leftarrow AKG(1^k)$

Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

ω = VESign(sk_A,apk,m)

$(pk_A, sk_A) \leftarrow KG(1^k)$

$(pk_B, sk_B) \leftarrow KG(1^k)$

$(apk, ask) \leftarrow AKG(1^k)$

Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

ω = VESign(sk_A,apk,m)

VEVerify(pk_A,apk,ω,m)

(pk_A,sk_A)$\leftarrow$KG(1^k)

(pk_B,sk_B)$\leftarrow$KG(1^k)

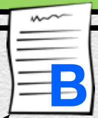
(apk,ask)$\leftarrow$AKG(1^k)

Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

$(pk_A, sk_A) \leftarrow KG(1^k)$

$\boxed{\omega = \mathrm{VESign}(sk_A, apk, m)}$

$\sigma =$ 

$= \mathrm{Sign}(sk_B, m)$

VEVerify(pk_A,apk,ω,m)

$(pk_B, sk_B) \leftarrow KG(1^k)$

$(apk, ask) \leftarrow AKG(1^k)$

Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

Verify(pk_B,σ,m)

ω = VESign(sk_A,apk,m)

VEVerify(pk_A,apk,ω,m)

$(pk_A,sk_A) \leftarrow KG(1^k)$

σ =  = Sign(sk_B,m)

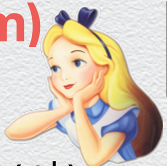
$(pk_B,sk_B) \leftarrow KG(1^k)$

$(apk,ask) \leftarrow AKG(1^k)$

Verifiably encrypted signatures [BGLS03]

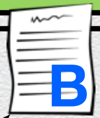
- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)

Verify(pk_B,σ,m)



$(pk_A,sk_A) \leftarrow KG(1^k)$

ω = VESign(sk_A,apk,m)

σ = 

= Sign(sk_B,m)

ω

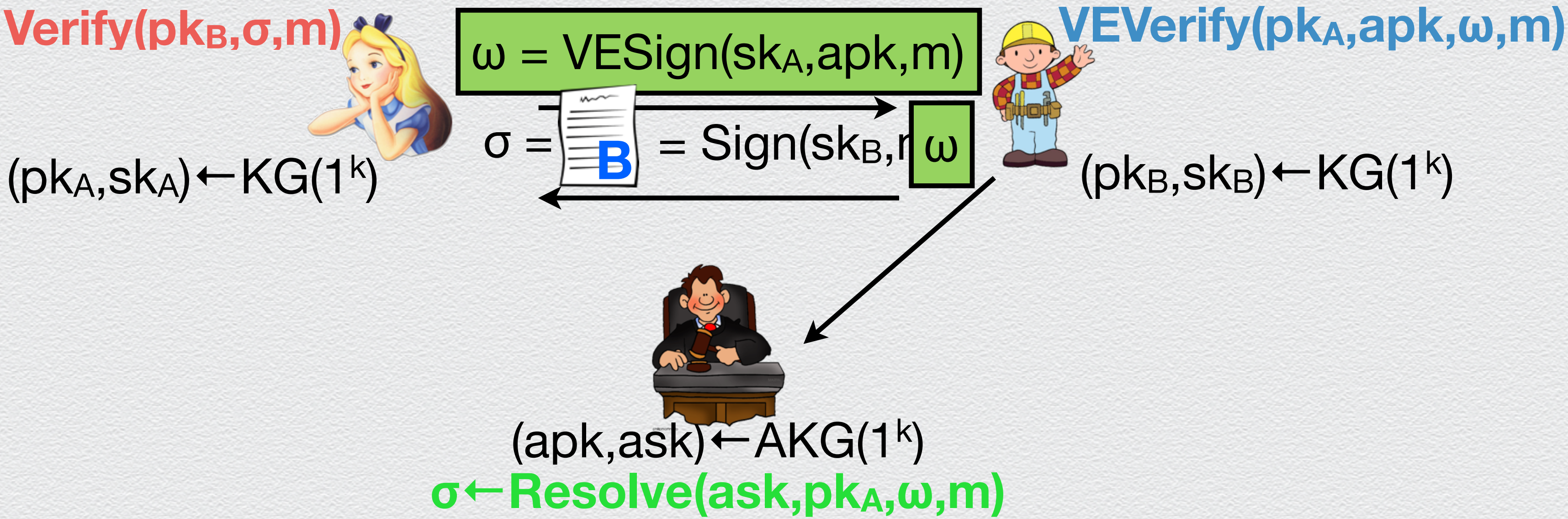
VEVerify(pk_A,apk,ω,m)

$(pk_B,sk_B) \leftarrow KG(1^k)$

$(apk,ask) \leftarrow AKG(1^k)$

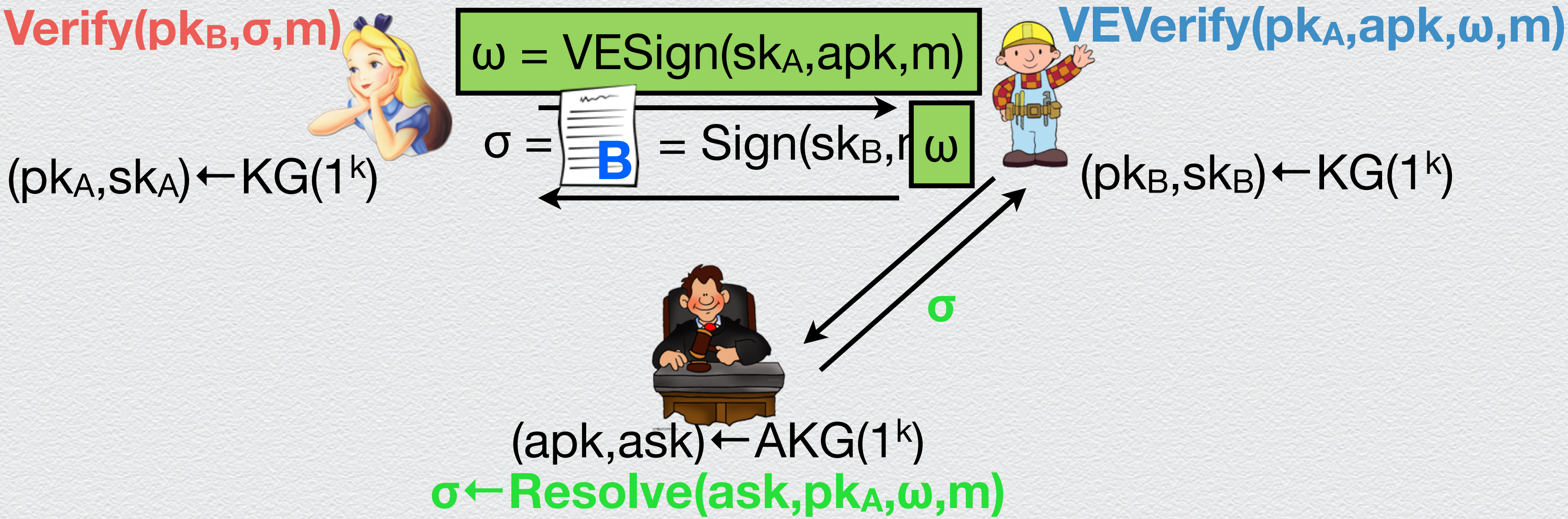
Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)



Verifiably encrypted signatures [BGLS03]

- A VES is a tuple (KG, Sign, Verify, AKG, VESign, VEVerify, Resolve)



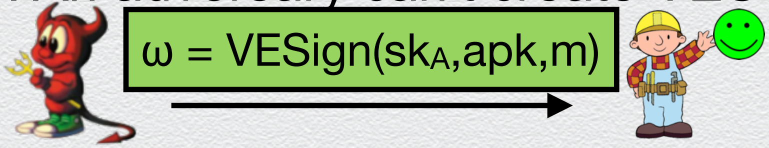
Verifiably encrypted signatures [BGLS03]

Verifiably encrypted signatures [BGLS03]

- A **secure** VES satisfies three properties:

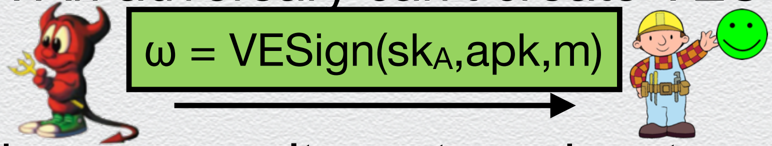
Verifiably encrypted signatures [BGLS03]

- A **secure** VES satisfies three properties:
  - **Unforgeability**: An adversary can't create VES

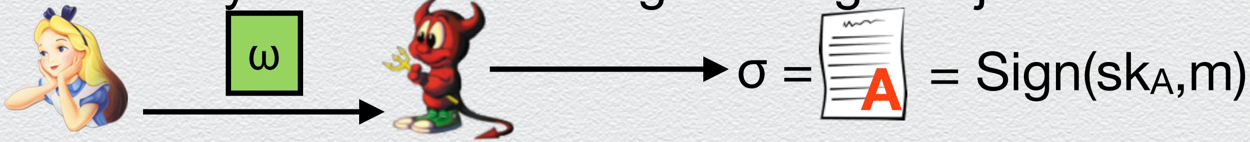


Verifiably encrypted signatures [BGLS03]

- A **secure** VES satisfies three properties:
  - **Unforgeability**: An adversary can't create VES

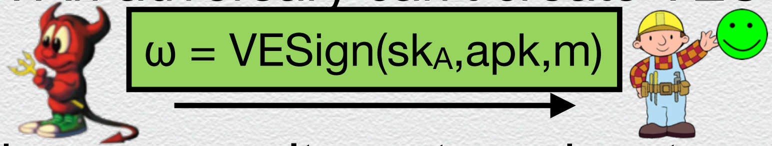


  - **Opacity**: An adversary can't create a signature given just VES

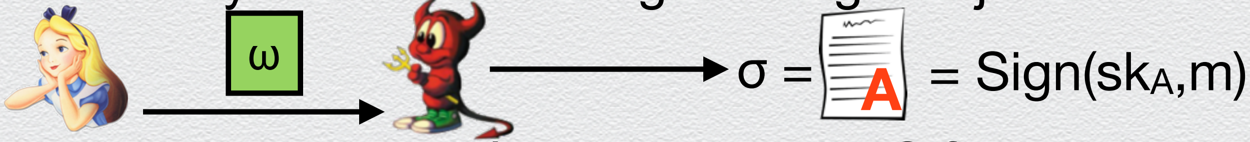


Verifiably encrypted signatures [BGLS03]

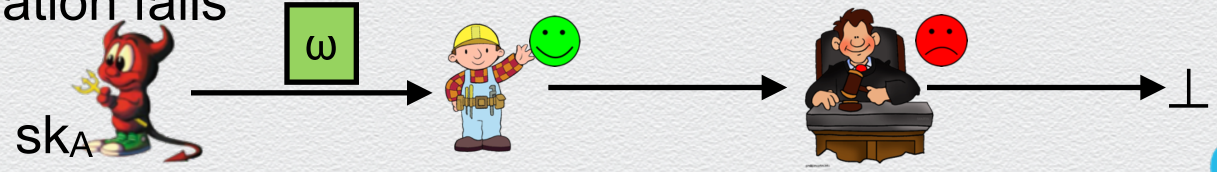
- A **secure** VES satisfies three properties:
  - **Unforgeability**: An adversary can't create VES



  - **Opacity**: An adversary can't create a signature given just VES



  - **Extractability**: An adversary can't create valid VES for which arbitration fails



A signature-based VES

Constructing VES with just signatures

Constructing VES with just signatures

- Assume we have a signature (KG′,Sign′,Verify′) with message space M′ and a transformation T from **(M,APK,0/1,Ω)** to M′

Constructing VES with just signatures

- Assume we have a signature (KG′,Sign′,Verify′) with message space M′ and a transformation T from **(M,APK,0/1,Ω)** to M′
- Sign, VESign, and Resolve all use Sign′, just sign different messages

Constructing VES with just signatures

- Assume we have a signature (KG′,Sign′,Verify′) with message space M′ and a transformation T from **(M,APK,0/1,Ω)** to M′
- Sign, VESign, and Resolve all use Sign′, just sign different messages

Sign VESign Resolve

Constructing VES with just signatures

- Assume we have a signature (KG′,Sign′,Verify′) with message space M′ and a transformation T from **(M,APK,0/1,Ω)** to M′
- Sign, VESign, and Resolve all use Sign′, just sign different messages

Sign VESign Resolve

T(m,⊥,⊥,⊥)

Constructing VES with just signatures

- Assume we have a signature (KG′,Sign′,Verify′) with message space M′ and a transformation T from **(M,APK,0/1,Ω)** to M′
- Sign, VESign, and Resolve all use Sign′, just sign different messages

Sign

T(m,⊥,⊥,⊥)

VESign

T(m,apk,0,⊥)

Resolve

Constructing VES with just signatures

- Assume we have a signature (KG′,Sign′,Verify′) with message space M′ and a transformation T from **(M,APK,0/1,Ω)** to M′
- Sign, VESign, and Resolve all use Sign′, just sign different messages

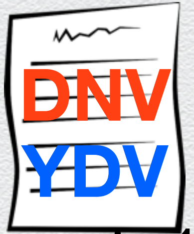
Sign

$T(m,\perp,\perp,\perp)$

VESign

$T(m,apk,0,\perp)$

Resolve



$T(m,apk,1,\omega)$

Constructing VES with just signatures

- Assume we have a signature (KG′,Sign′,Verify′) with message space M′ and a transformation T from **(M,APK,0/1,Ω)** to M′
- Sign, VESign, and Resolve all use Sign′, just sign different messages

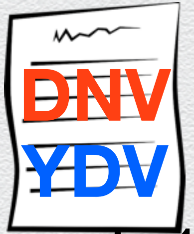
Sign

$T(m,\perp,\perp,\perp)$

VESign

$T(m,apk,0,\perp)$

Resolve



$T(m,apk,1,\omega)$

- There are **two signatures**, and Verify checks for both

Security of signature-based construction

Security of signature-based construction

- Unforgeability: can't create VES

Security of signature-based construction

- Unforgeability: can't create VES

$T(m, apk, 0, \perp)$

Security of signature-based construction

- Unforgeability: can't create VES
- Opacity: can't create signature given VES

$T(m, apk, 0, \perp)$

Security of signature-based construction

- Unforgeability: can’t create VES
- Opacity: can’t create signature given VES

$T(m,apk,0,\perp)$

$T(m,apk,0,\perp)$

Security of signature-based construction

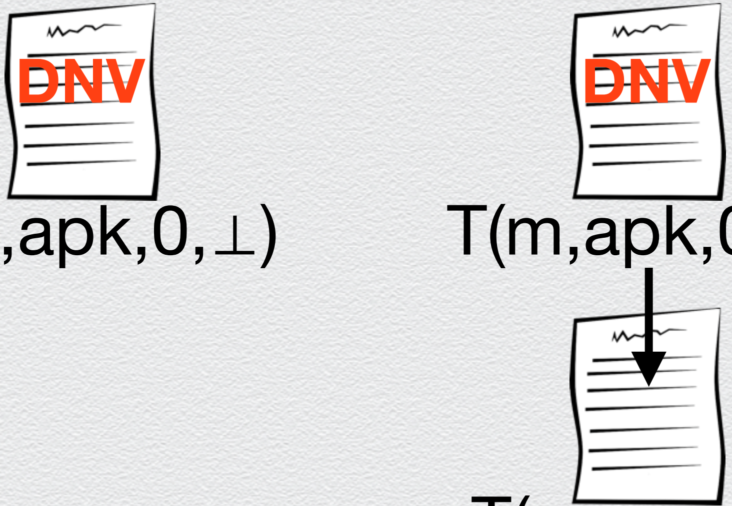
- Unforgeability: can't create VES
- Opacity: can't create signature given VES

$T(m, apk, 0, \perp)$

$T(m, apk, 0, \perp)$

Security of signature-based construction

- Unforgeability: can't create VES
- Opacity: can't create signature given VES

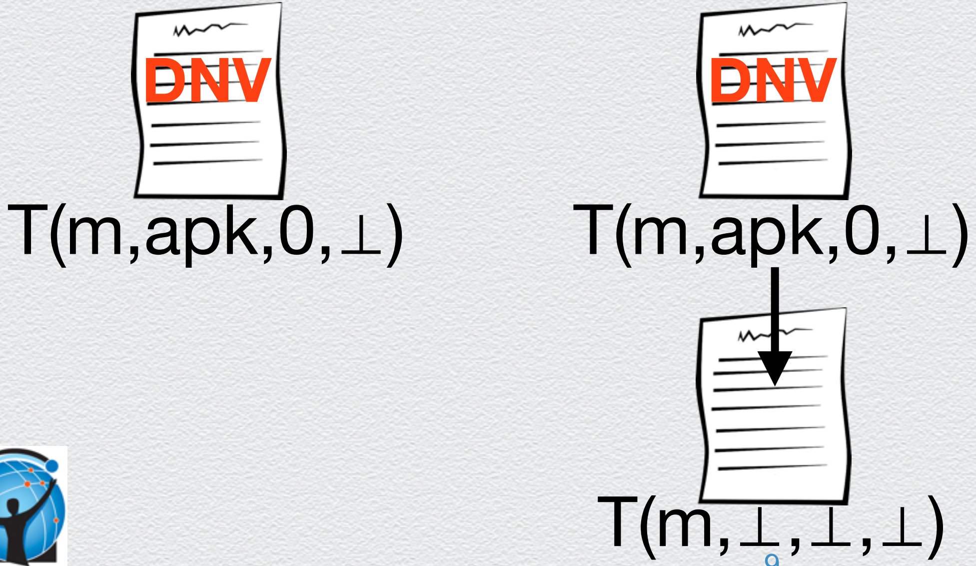


$T(m, apk, 0, \perp)$ $T(m, apk, 0, \perp)$

$T(m, \perp, \perp, \perp)$

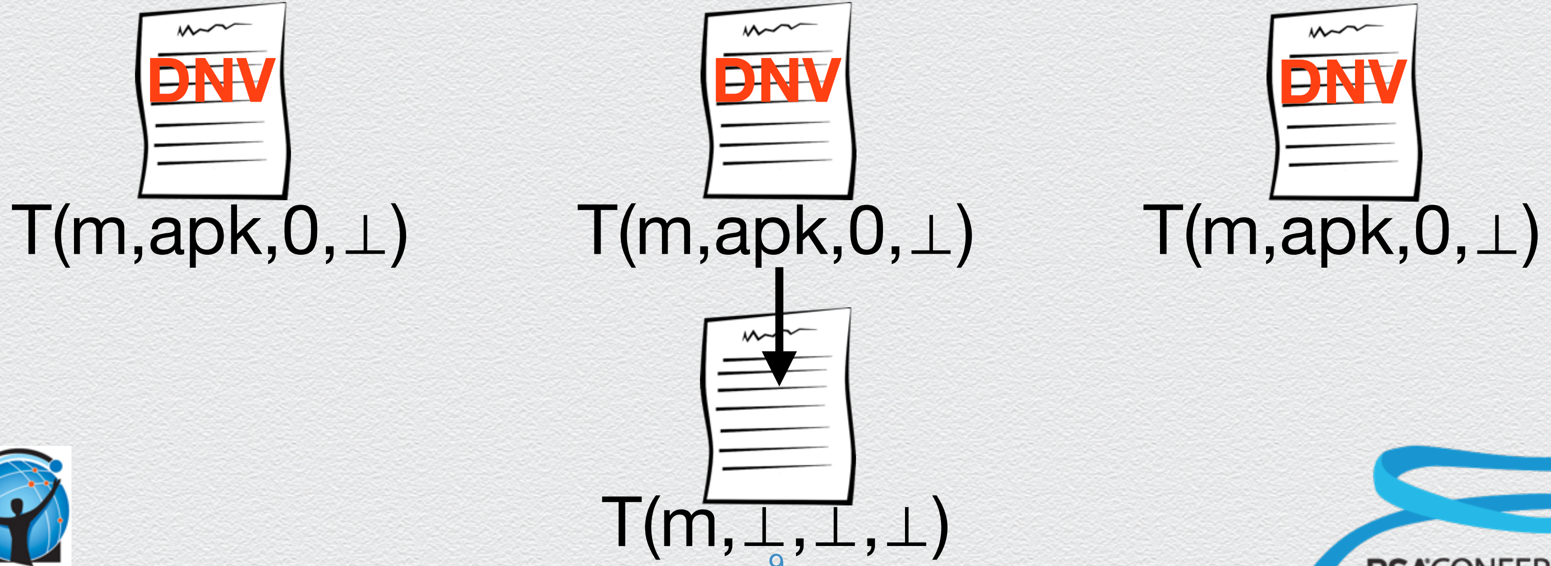
Security of signature-based construction

- Unforgeability: can't create VES
- Opacity: can't create signature given VES
- Extractability: can't create VES for which arbitration fails



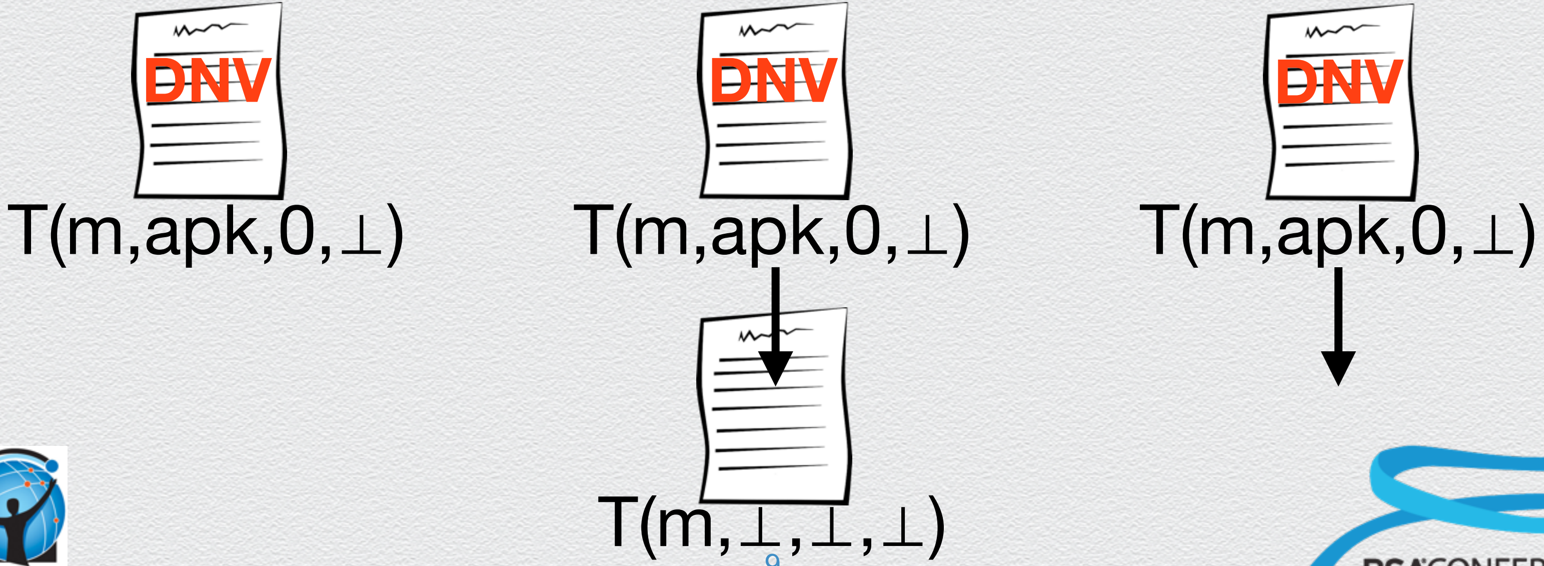
Security of signature-based construction

- Unforgeability: can't create VES
- Opacity: can't create signature given VES
- Extractability: can't create VES for which arbitration fails



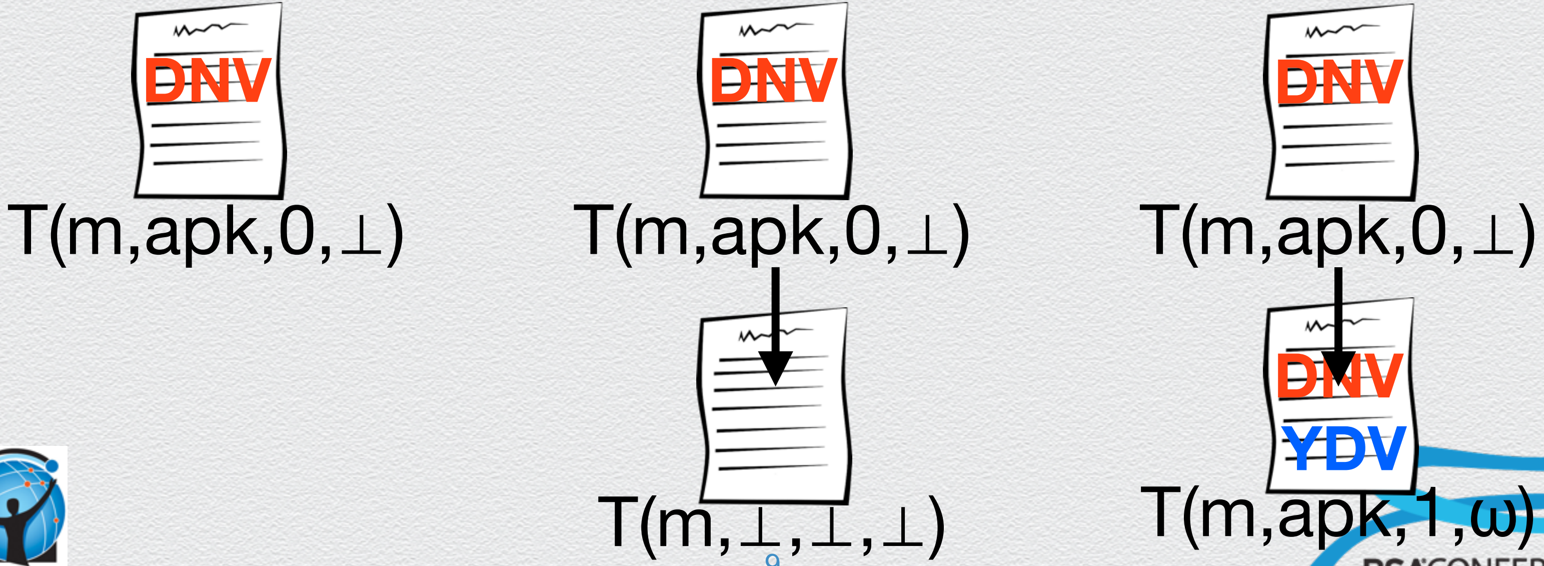
Security of signature-based construction

- Unforgeability: can't create VES
- Opacity: can't create signature given VES
- Extractability: can't create VES for which arbitration fails



Security of signature-based construction

- Unforgeability: can't create VES
- Opacity: can't create signature given VES
- Extractability: can't create VES for which arbitration fails



Resolution independence

Resolution independence

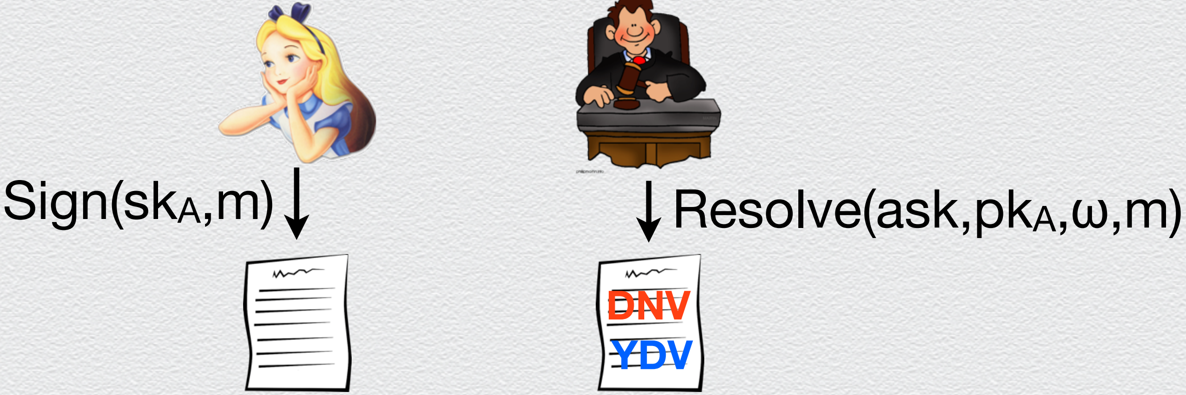
- The problem with the signature-based construction: Bob got a different object from Alice than from the arbiter!

Resolution independence

- The problem with the signature-based construction: Bob got a different object from Alice than from the arbiter!

Resolution independence

- The problem with the signature-based construction: Bob got a different object from Alice than from the arbiter!

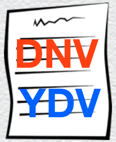


Resolution independence

- The problem with the signature-based construction: Bob got a different object from Alice than from the arbiter!

Sign(sk_A,m) ↓

↓ Resolve(ask,pk_A,ω,m)



- **Resolution independence**: the distributions {Sign(sk,m)} and {Resolve(ask,pk,ω,m)} are identical

Separating our construction from existing ones

Separating our construction from existing ones

- Signature construction is not resolution independent: **σ vs. (apk,ω,ω′)**

Separating our construction from existing ones

- Signature construction is not resolution independent: **σ vs. (apk,ω,ω')**

- But it is satisfied by all existing VES constructions
  - [BGLS03] uses bilinear groups, BLS signatures, deterministic Resolve
  - [LOSSW05] uses bilinear groups, Waters signatures, randomized Resolve
  - [R09] uses RSA groups and signatures, deterministic Resolve

Resolution duplication

Resolution duplication

- Verifiably **encrypted** signatures: encryption really must be happening

Resolution duplication

- Verifiably **encrypted** signatures: encryption really must be happening
- Can form **ω** so that no one can extract **σ** from **ω** (by **opacity**), except the arbiter can extract **σ′** from the same distribution (by **resolution independence**)

Resolution duplication

- Verifiably **encrypted** signatures: encryption really must be happening
- Can form **ω** so that no one can extract **σ** from **ω** (by **opacity**), except the arbiter can extract **σ′** from the same distribution (by **resolution independence**)
- Not quite encryption: **σ′ might be different from σ**

Resolution duplication

- Verifiably **encrypted** signatures: encryption really must be happening
- Can form **ω** so that no one can extract **σ** from **ω** (by **opacity**), except the arbiter can extract **σ′** from the same distribution (by **resolution independence**)
- Not quite encryption: **σ′ might be different from σ**
- **Resolution duplication** requires: (1) resolution independence, (2) deterministic Resolve, and (3) that there exists an algorithm Extract such that Extract(sk,m,r) = Resolve(ask,pk,VESign(sk,apk,m;r),m)

Constructing PKE with resolution duplication

Constructing PKE with resolution duplication

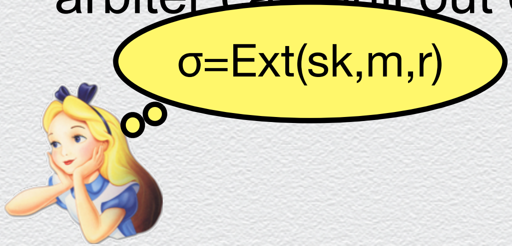
- With resolution duplication, Alice can form ω := VESign(sk,apk,m;r) so that no one can pull out Sign(sk,m) from ω (by opacity), except the arbiter can pull out σ, and Alice can duplicate σ using Extract(sk,m,r)

Constructing PKE with resolution duplication

- With resolution duplication, Alice can form ω := VESign(sk,apk,m;r) so that no one can pull out Sign(sk,m) from ω (by opacity), except the arbiter can pull out σ, and Alice can duplicate σ using Extract(sk,m,r)

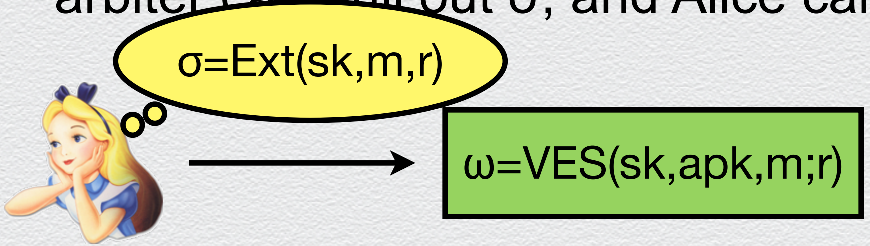
Constructing PKE with resolution duplication

- With resolution duplication, Alice can form $\omega := \mathrm{VESign}(sk,apk,m;r)$ so that no one can pull out $\mathrm{Sign}(sk,m)$ from ω (by opacity), except the arbiter can pull out σ, and Alice can duplicate σ using $\mathrm{Extract}(sk,m,r)$



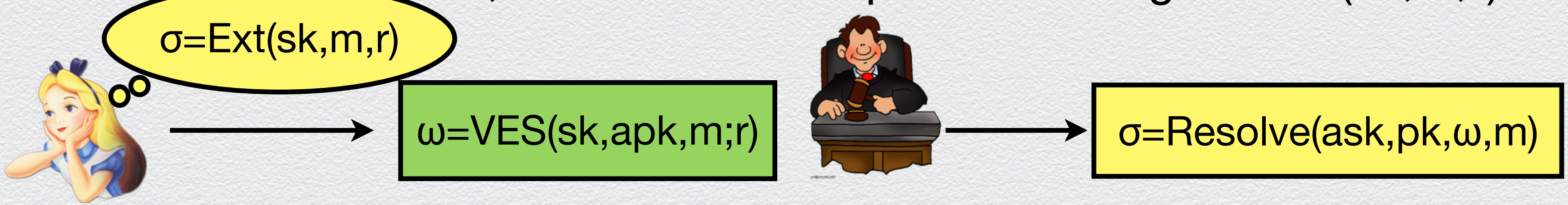
Constructing PKE with resolution duplication

- With resolution duplication, Alice can form ω := VESign(sk,apk,m;r) so that no one can pull out Sign(sk,m) from ω (by opacity), except the arbiter can pull out σ, and Alice can duplicate σ using Extract(sk,m,r)



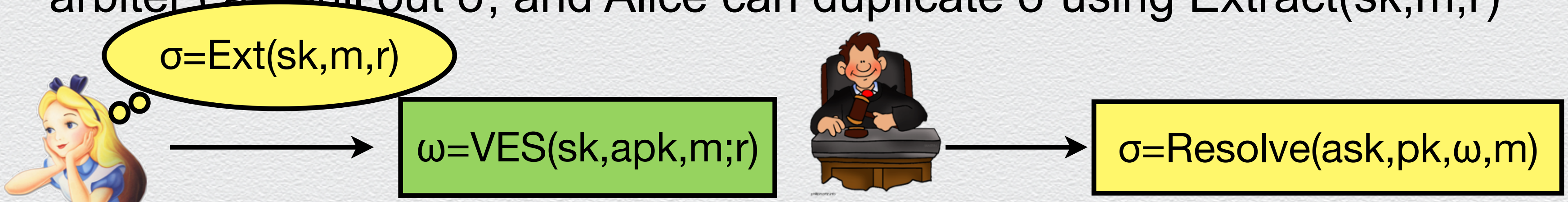
Constructing PKE with resolution duplication

- With resolution duplication, Alice can form ω := VESign(sk,apk,m;r) so that no one can pull out Sign(sk,m) from ω (by opacity), except the arbiter can pull out σ, and Alice can duplicate σ using Extract(sk,m,r)



Constructing PKE with resolution duplication

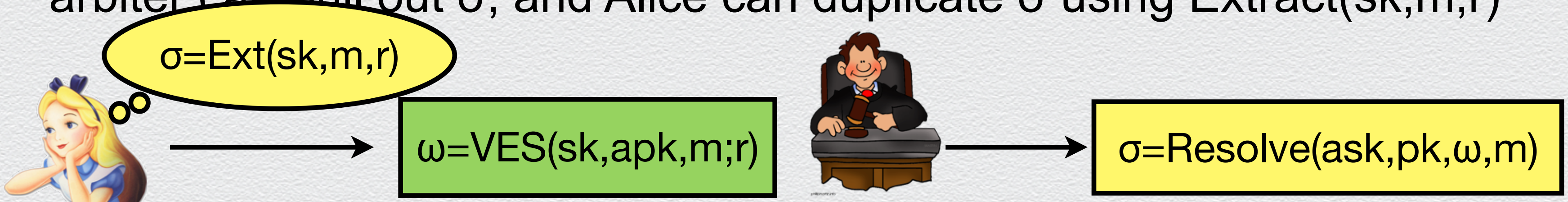
- With resolution duplication, Alice can form ω := VESign(sk,apk,m;r) so that no one can pull out Sign(sk,m) from ω (by opacity), except the arbiter can pull out σ, and Alice can duplicate σ using Extract(sk,m,r)



- This lets us “encrypt” signatures, but we want to encrypt arbitrary bits

Constructing PKE with resolution duplication

- With resolution duplication, Alice can form $\omega := \mathrm{VESign}(sk,apk,m;r)$ so that no one can pull out $\mathrm{Sign}(sk,m)$ from ω (by opacity), except the arbiter can pull out σ, and Alice can duplicate σ using $\mathrm{Extract}(sk,m,r)$



- This lets us “encrypt” signatures, but we want to encrypt arbitrary bits
- Adapt Goldreich-Levin trick [GL89]; show that it is hard to predict (compute) **$\langle\sigma,r\rangle = \sum \sigma_i \cdot r_i \bmod 2$** given just ω and r

Constructing PKE with resolution duplication

Constructing PKE with resolution duplication

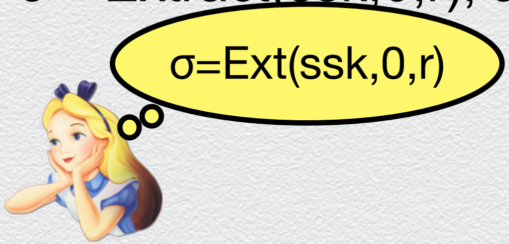
- EKeyGen(1k): Output (pk,sk)$\leftarrow$AKG(1^k) 

Constructing PKE with resolution duplication

- EKeyGen(1k): Output $(pk,sk) \leftarrow AKG(1^k)$ 
- Enc(pk,m): Generate $(spk,ssk) \leftarrow KG(1^k)$, $\omega \leftarrow VESign(ssk,pk,0;r)$, $\sigma \leftarrow Extract(ssk,0,r)$, and $r_\sigma \leftarrow \{0,1\}^{|\sigma|}$. Output $c = (spk, \omega, r_\sigma, m \oplus \langle \sigma, r_\sigma \rangle)$

Constructing PKE with resolution duplication

- EKeyGen(1k): Output $(pk,sk) \leftarrow AKG(1^k)$ 
- Enc(pk,m): Generate $(spk,ssk) \leftarrow KG(1^k)$, $\omega \leftarrow VESign(ssk,pk,0;r)$, $\sigma \leftarrow Extract(ssk,0,r)$, and $r_\sigma \leftarrow \{0,1\}^{|\sigma|}$. Output $c = (spk, \omega, r_\sigma, m \oplus \langle\sigma, r_\sigma\rangle)$



Constructing PKE with resolution duplication

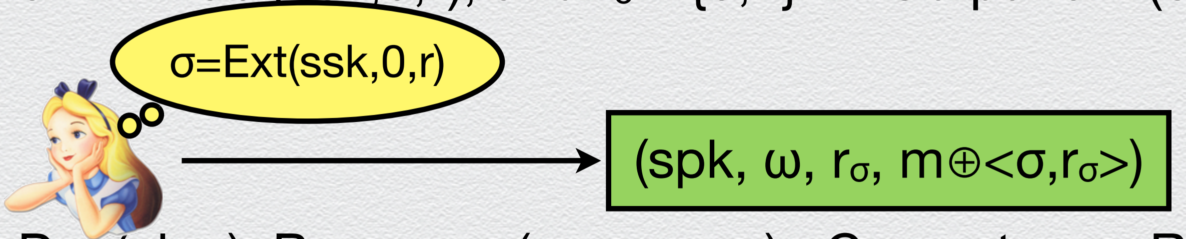
- EKeyGen(1k): Output $(pk,sk) \leftarrow AKG(1^k)$ 
- Enc(pk,m): Generate $(spk,ssk) \leftarrow KG(1^k)$, $\omega \leftarrow VESign(ssk,pk,0;r)$, $\sigma \leftarrow Extract(ssk,0,r)$, and $r_\sigma \leftarrow \{0,1\}^{|\sigma|}$. Output $c = (spk, \omega, r_\sigma, m \oplus \langle \sigma, r_\sigma \rangle)$

$\sigma = Ext(ssk,0,r)$

$\longrightarrow$ $\boxed{(spk, \omega, r_\sigma, m \oplus \langle \sigma, r_\sigma \rangle)}$

Constructing PKE with resolution duplication

- EKeyGen(1k): Output $(pk,sk) \leftarrow AKG(1^k)$ 
- Enc(pk,m): Generate $(spk,ssk) \leftarrow KG(1^k)$, $\omega \leftarrow VESign(ssk,pk,0;r)$, $\sigma \leftarrow Extract(ssk,0,r)$, and $r_\sigma \leftarrow \{0,1\}^{|\sigma|}$. Output $c = (spk,\ \omega,\ r_\sigma,\ m \oplus \langle\sigma,r_\sigma\rangle)$



- Dec(sk,c): Parse $c = (c_1,c_2,c_3,c_4)$. Compute $\sigma = Resolve(sk,c_1,c_2,0)$ and output $c_4 \oplus \langle\sigma,c_3\rangle$

Constructing PKE with resolution duplication

- EKeyGen(1k): Output (pk,sk)$\leftarrow$AKG(1^k)

- Enc(pk,m): Generate (spk,ssk)$\leftarrow$KG(1^k), $\omega\leftarrow$VESign(ssk,pk,0;r), $\sigma\leftarrow$Extract(ssk,0,r), and $r_\sigma\leftarrow\{0,1\}^{|\sigma|}$. Output c = (spk, ω, r_σ, m$\oplus$<σ,r_σ>)

σ=Ext(ssk,0,r)

(spk, ω, r_σ, m$\oplus$<σ,r_σ>)

- Dec(sk,c): Parse c = (c_1,c_2,c_3,c_4). Compute σ = Resolve(sk,c_1,c_2,0) and output $c_4\oplus$<σ,c_3>

σ=Resolve(ask,pk,ω,0)

Constructing PKE with resolution duplication

- EKeyGen(1k): Output $(pk,sk) \leftarrow AKG(1^k)$ 
- Enc(pk,m): Generate $(spk,ssk) \leftarrow KG(1^k)$, $\omega \leftarrow VESign(ssk,pk,0;r)$, $\sigma \leftarrow Extract(ssk,0,r)$, and $r_\sigma \leftarrow \{0,1\}^{|\sigma|}$. Output $c = (spk, \omega, r_\sigma, m \oplus \langle\sigma, r_\sigma\rangle)$

$\sigma = Ext(ssk,0,r)$

$\longrightarrow$ $\boxed{(spk, \omega, r_\sigma, m \oplus \langle\sigma, r_\sigma\rangle)}$

- Dec(sk,c): Parse $c = (c_1,c_2,c_3,c_4)$. Compute $\sigma = Resolve(sk,c_1,c_2,0)$ and output $c_4 \oplus \langle\sigma, c_3\rangle$

$\longrightarrow$ $\boxed{\sigma = Resolve(ask,pk,\omega,0)}$ $\boxed{c_4 \oplus \langle\sigma,c_3\rangle = m \oplus \langle\sigma,r_\sigma\rangle \oplus \langle\sigma,c_3\rangle = m}$

Constructing PKE with resolution duplication

- EKeyGen(1k): Output $(pk,sk) \leftarrow AKG(1^k)$

- Enc(pk,m): Generate $(spk,ssk) \leftarrow KG(1^k)$, $\omega \leftarrow VESign(ssk,pk,0;r)$, $\sigma \leftarrow Extract(ssk,0,r)$, and $r_\sigma \leftarrow \{0,1\}^{|\sigma|}$. Output $c = (spk, \omega, r_\sigma, m \oplus \langle \sigma, r_\sigma \rangle)$

σ=Ext(ssk,0,r)

$\rightarrow \boxed{(spk, \omega, r_\sigma, m \oplus \langle \sigma, r_\sigma \rangle)}$

The same by resolution duplication!

- Dec(sk,c): Parse $c = (c_1,c_2,c_3,c_4)$. Compute σ = Resolve$(sk,c_1,c_2,0)$ and output $c_4 \oplus \langle \sigma, c_3 \rangle$

$\rightarrow \boxed{\sigma = Resolve(ask,pk,\omega,0)}$ $\boxed{c_4 \oplus \langle \sigma, c_3 \rangle = m \oplus \langle \sigma, r_\sigma \rangle \oplus \langle \sigma, c_3 \rangle = m}$

Constructing PKE with resolution duplication

Constructing PKE with resolution duplication

- Interestingly, resolution duplication contributed to the **correctness** of the encryption scheme rather than its security

$$\boxed{c_4 \oplus \langle \sigma, c_3 \rangle = m \oplus \langle \sigma, r_\sigma \rangle \oplus \langle \sigma, c_3 \rangle = m}$$

Constructing PKE with resolution duplication

- Interestingly, resolution duplication contributed to the **correctness** of the encryption scheme rather than its security

$$\boxed{c_4 \oplus \langle \sigma, c_3 \rangle = m \oplus \langle \sigma, r_\sigma \rangle \oplus \langle \sigma, c_3 \rangle = m}$$

- **IND-CPA security** follows fairly directly from opacity

RSACONFERENCE2014
FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Conclusions

Conclusions and open problems

Conclusions and open problems

- Existing VES definitions might not capture desired functionality

Conclusions and open problems

- Existing VES definitions might not capture desired functionality
  - Provided a solely signature-based VES

Conclusions and open problems

- Existing VES definitions might not capture desired functionality
  - Provided a solely signature-based VES
  - Defined resolution independence to “separate” this construction from existing ones

Conclusions and open problems

- Existing VES definitions might not capture desired functionality
  - Provided a solely signature-based VES
  - Defined resolution independence to "separate" this construction from existing ones
  - Demonstrated how stronger resolution duplication could be used to construct public-key encryption

Conclusions and open problems

- Existing VES definitions might not capture desired functionality
  - Provided a solely signature-based VES
  - Defined resolution independence to "separate" this construction from existing ones
  - Demonstrated how stronger resolution duplication could be used to construct public-key encryption
- Are VES just misnamed? Or would applications fail if encryption part were missing?