Bi-Deniable Public-Key Encryption

Adam O'Neill[1,2] Chris Peikert[1] Brent Waters[2]

[1]Georgia Tech

[2]U Texas, Austin

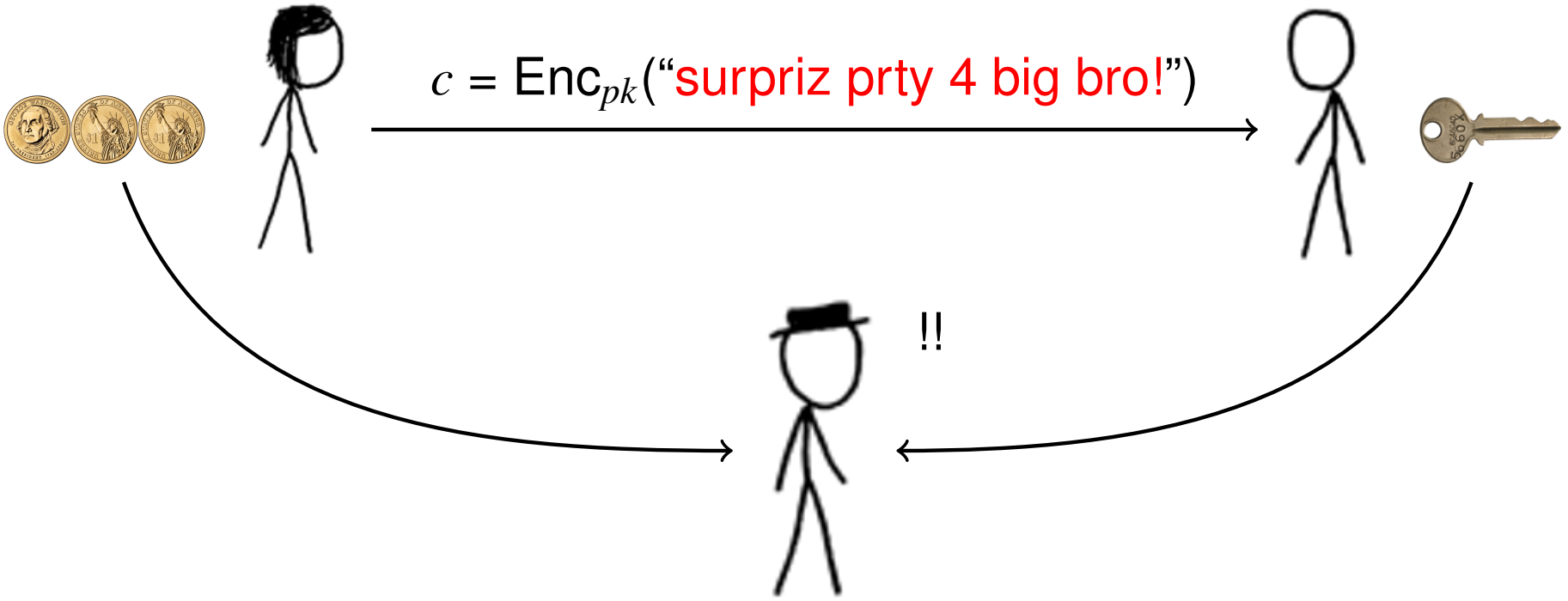
CRYPTO 2011
17 Aug

Deniable Encryption [CDNO'97]

$c = \mathrm{Enc}_{pk}(\text{“surpriz prty 4 big bro!”})$

(Images courtesy xkcd.org)

Deniable Encryption [CDNO'97]



(Images courtesy xkcd.org)

Deniable Encryption [CDNO'97]

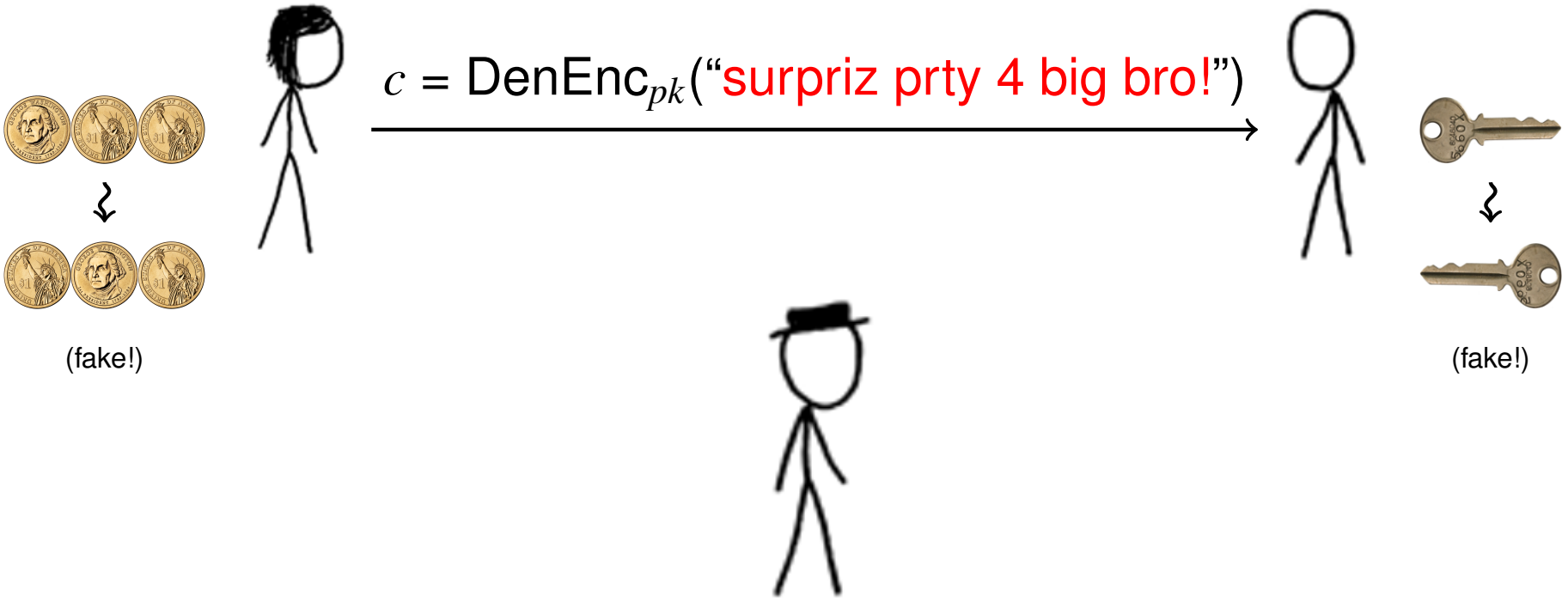
$c = \mathrm{DenEnc}_{pk}(\text{“surpriz prty 4 big bro!”})$

What We Want: Bi-Deniability

1. Bob decrypts Alice's message correctly, but . . .

(Images courtesy xkcd.org)

Deniable Encryption [CDNO'97]

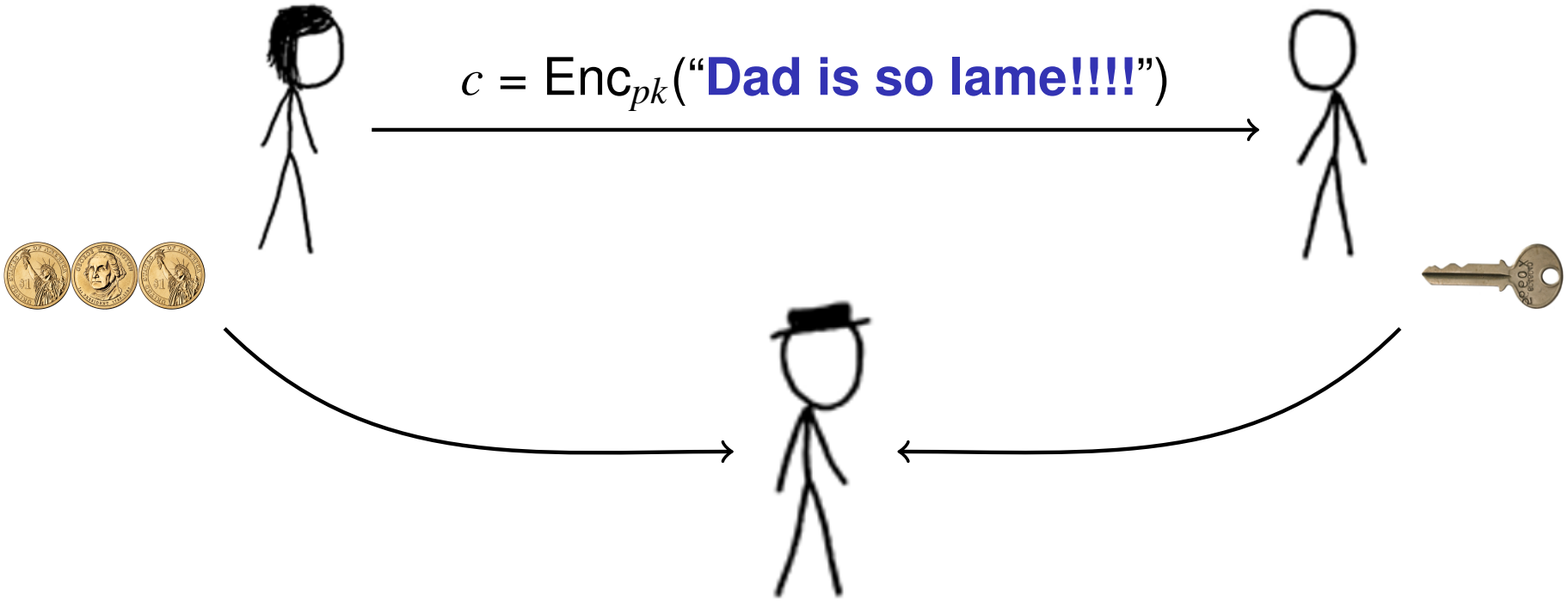


What We Want: Bi-Deniability

1. Bob decrypts Alice's message correctly, but . . .

(Images courtesy xkcd.org)

Deniable Encryption [CDNO'97]

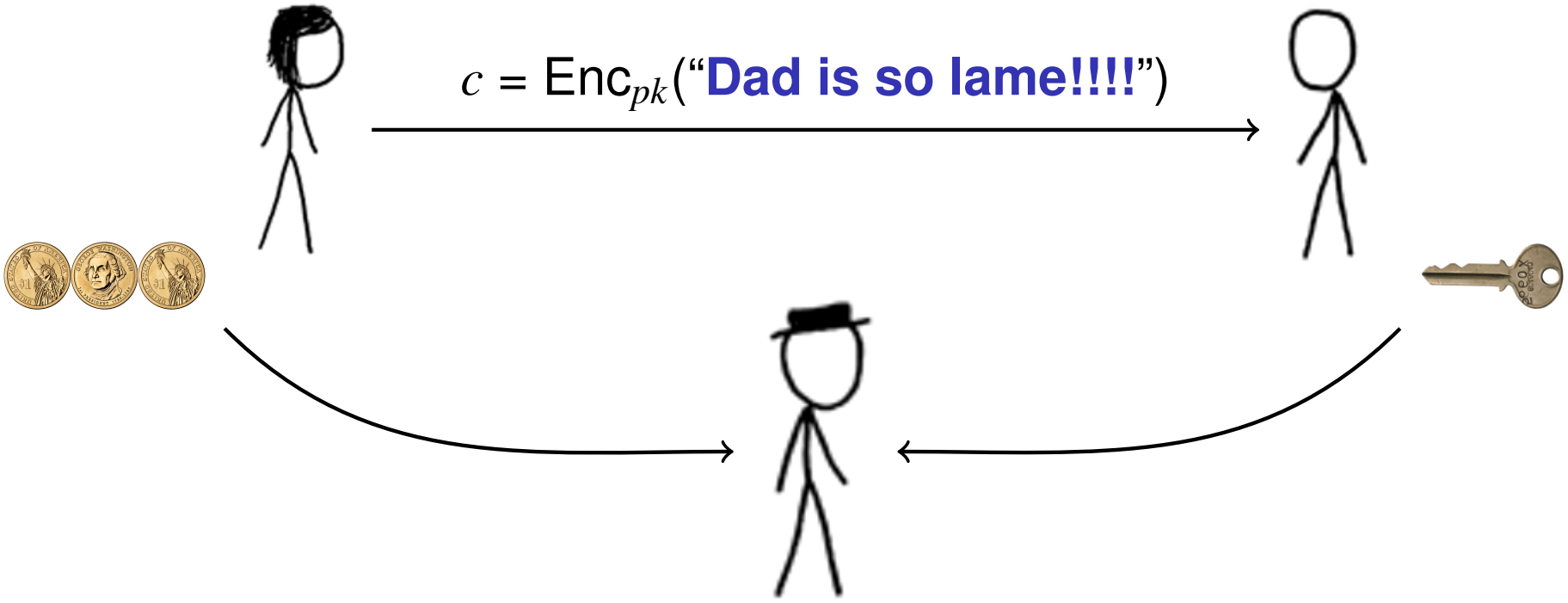


What We Want: Bi-Deniability

1. Bob decrypts Alice's message correctly, but . . .
2. Fake coins & keys "look as if" another message was encrypted.

(Images courtesy xkcd.org)

Deniable Encryption [CDNO'97]



What We Want: Bi-Deniability

1. Bob decrypts Alice's message correctly, but . . .
2. Fake coins & keys "look as if" another message was encrypted.

⋆⋆ Coercion is after the fact (cf. "uncoercible communication" [BT'94])

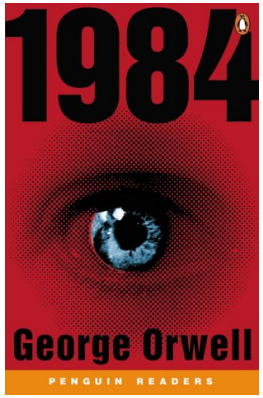
(Images courtesy xkcd.org)

Applications of Deniability

1. Anti-coercion: journalists, lawyers, whistle-blowers

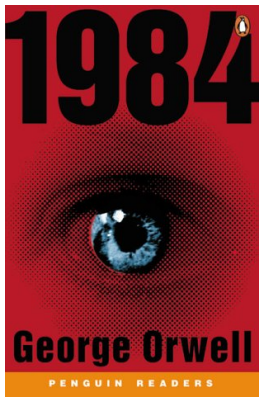
Applications of Deniability

1. Anti-coercion: journalists, lawyers, whistle-blowers



Applications of Deniability

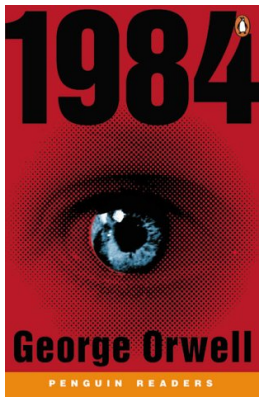
1. Anti-coercion: journalists, lawyers, whistle-blowers



2. Voting (?): can reveal *any* candidate, so can't 'sell' vote

Applications of Deniability

1. Anti-coercion: journalists, lawyers, whistle-blowers

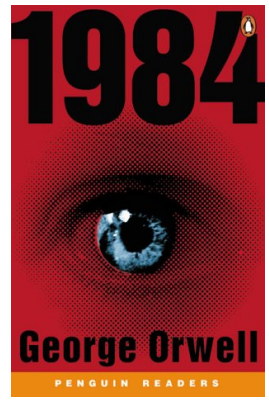


2. Voting (?): can reveal *any* candidate, so can't 'sell' vote

3. Implies selective-opening security [DNRS'99,BHY'09]

Applications of Deniability

1. Anti-coercion: journalists, lawyers, whistle-blowers



2. Voting (?): can reveal *any* candidate, so can't 'sell' vote

3. Implies selective-opening security [DNRS'99,BHY'09]

4. Implies noncommitting encryption for adaptive corruption [CFGN'96]

Prior Work

Theory [CDNO'97]

- Sender-deniable public-key encryption
- Receiver-deniability with interaction
- Bi-deniability via interaction w/ 3rd parties (one must remain uncoerced)

Prior Work

Theory [CDNO'97]

- Sender-deniable public-key encryption
- Receiver-deniability with interaction
- Bi-deniability via interaction w/ 3rd parties (one must remain uncoerced)

Practice: TrueCrypt, Rubberhose FS, . . .

- "Plausible deniability:" *move along, no message here. . .*

  Maybe OK for *storage*, but not so much for *communication*.

This Work

1. Bi-deniable encryption: sender & receiver are *simultaneously* coercible, and can reveal any message (chosen at coercion time).

This Work

1. Bi-deniable encryption: sender & receiver are *simultaneously* coercible, and can reveal any message (chosen at coercion time).

   Works in "multi-distributional" (flexible) model: DenGen & DenEnc algorithms, equivocated as if Gen & Enc were run.

This Work

1. Bi-deniable encryption: sender & receiver are *simultaneously* coercible, and can reveal any message (chosen at coercion time).

   Works in "multi-distributional" (flexible) model: DenGen & DenEnc algorithms, equivocated as if Gen & Enc were run.

   - ⋆ True public-key schemes: non-interactive, no 3rd parties
   - ⋆ One generic construction [DN'00] & one using lattices [GPV'08]
   - ⋆ Both have $|\text{keys}| > |\text{messages}|$ . . . but this is inherent [Nielsen'02]

This Work

1. Bi-deniable encryption: sender & receiver are *simultaneously* coercible, and can reveal any message (chosen at coercion time).

   Works in "multi-distributional" (flexible) model: DenGen & DenEnc algorithms, equivocated as if Gen & Enc were run.

   - ⋆ True public-key schemes: non-interactive, no 3rd parties
   - ⋆ One generic construction [DN'00] & one using lattices [GPV'08]
   - ⋆ Both have $|\text{keys}| > |\text{messages}|$ . . . but this is inherent [Nielsen'02]

2. "Plan-ahead" bi-deniability with short keys

   (analogue of "somewhat non-committing" encryption [GWZ'09])

   - ⋆ Bounded number of alternative messages, decided in advance
   - ⋆ Sender & receiver automatically agree on fake message

This Work

1. Bi-deniable encryption: sender & receiver are *simultaneously* coercible, and can reveal any message (chosen at coercion time).

   Works in "multi-distributional" (flexible) model: DenGen & DenEnc algorithms, equivocated as if Gen & Enc were run.

   - ⋆ True public-key schemes: non-interactive, no 3rd parties
   - ⋆ One generic construction [DN'00] & one using lattices [GPV'08]
   - ⋆ Both have $|\text{keys}| > |\text{messages}|$ . . . but this is inherent [Nielsen'02]

2. "Plan-ahead" bi-deniability with short keys

   (analogue of "somewhat non-committing" encryption [GWZ'09])

   - ⋆ Bounded number of alternative messages, decided in advance
   - ⋆ Sender & receiver automatically agree on fake message

3. Analogous solutions in the ID-based setting.

Subsequent Work

1. [DF'11] announced interactive, fully sender-deniable encryption

Subsequent Work

1. [DF'11] announced interactive, fully sender-deniable encryption
   - ⋆ Unfortunately, there is a fatal bug in deniability claim (& an attack)
   - ⋆ Obtaining full deniability remains an intriguing open problem!

Subsequent Work

1. [DF'11] announced interactive, fully sender-deniable encryption
   - ⋆ Unfortunately, there is a fatal bug in deniability claim (& an attack)
   - ⋆ Obtaining full deniability remains an intriguing open problem!

2. "Fully receiver-/bi-deniable PKE is impossible" [BNNO'11]
   - ⋆ Formally: σ-bit secret key $\Rightarrow$ $(1/\sigma)$-distinguishable real vs. fake
   - ⋆ Don't deny the impossibility — instead, be "flexible."

“Flexible” Bi-Deniability

- ‘Normal’ Gen, Enc, Dec algorithms . . .
  plus ‘deniable’ DenGen, DenEnc and ‘faking’ RecFake, SendFake.

“Flexible” Bi-Deniability

- ‘Normal’ Gen, Enc, Dec algorithms . . .
  plus ‘deniable’ DenGen, DenEnc and ‘faking’ RecFake, SendFake.

- The following are indistinguishable for all bits b, b':

$(pk, sk) \leftarrow \mathsf{Gen}$ $c \leftarrow \mathsf{Enc}(pk, b; r)$	
View: (pk, c, sk, r)	

“Flexible” Bi-Deniability

- ‘Normal’ Gen, Enc, Dec algorithms . . .
  plus ‘deniable’ DenGen, DenEnc and ‘faking’ RecFake, SendFake.

- The following are indistinguishable for all bits b, b':

$(pk, sk) \leftarrow \mathsf{Gen}$	$(pk, fk) \leftarrow \mathsf{DenGen}$
$c \leftarrow \mathsf{Enc}(pk, b; r)$	$c \leftarrow \mathsf{DenEnc}(pk, b'; r)$
	- - - - - - - - - -
	$sk^* \leftarrow \mathsf{RecFake}(fk, c, b)$
	$r^* \leftarrow \mathsf{SendFake}(pk, r, b', b)$
View: (pk, c, sk, r)	View: (pk, c, sk^*, r^*)

“Flexible” Bi-Deniability

- ‘Normal’ Gen, Enc, Dec algorithms . . .
  plus ‘deniable’ DenGen, DenEnc and ‘faking’ RecFake, SendFake.

- The following are indistinguishable for all bits b, b':

$(pk, sk) \leftarrow \mathsf{Gen}$	$(pk, fk) \leftarrow \mathsf{DenGen}$
$c \leftarrow \mathsf{Enc}(pk, b; r)$	$c \leftarrow \mathsf{DenEnc}(pk, b'; r)$
	- - - - - - - - - -
	$sk^* \leftarrow \mathsf{RecFake}(fk, c, b)$
	$r^* \leftarrow \mathsf{SendFake}(pk, r, b', b)$
View: (pk, c, sk, r)	View: (pk, c, sk^*, r^*)

(Even better, RecFake could output fake *coins* for Gen, instead of sk^*.)

"Flexible" Bi-Deniability

- 'Normal' Gen, Enc, Dec algorithms . . .
  plus 'deniable' DenGen, DenEnc and 'faking' RecFake, SendFake.

- The following are indistinguishable for all bits b, b':

$(pk, sk) \leftarrow \mathsf{Gen}$	$(pk, fk) \leftarrow \mathsf{DenGen}$
$c \leftarrow \mathsf{Enc}(pk, b; r)$	$c \leftarrow \mathsf{DenEnc}(pk, b'; r)$
	- - - - - - - - - -
	$sk^* \leftarrow \mathsf{RecFake}(fk, c, b)$
	$r^* \leftarrow \mathsf{SendFake}(pk, r, b', b)$
View: (pk, c, sk, r)	View: (pk, c, sk^*, r^*)

(Even better, RecFake could output fake *coins* for Gen, instead of sk^*.)

- "Full" deniability requires equivocable Gen and Enc algs.

Is (Flexible) Deniability Meaningful?

Objection #1

- Everyone knows that the coins & message could be fake.

So who do we think we're fooling?

Is (Flexible) Deniability Meaningful?

Objection #1

- Everyone knows that the coins & message could be fake.

So who do we think we're fooling?

Answer

- 'Perfectly secret' communication is inherently deniable...

...but most encryption introduces risk of coercion!

Is (Flexible) Deniability Meaningful?

Objection #1

- Everyone knows that the coins & message could be fake.

So who do we think we're fooling?

Answer

- 'Perfectly secret' communication is inherently deniable...

...but most encryption introduces risk of coercion!

- Deniable encryption avoids this side-effect risk.

Is (Flexible) Deniability Meaningful?

Objection #1

- Everyone knows that the coins & message could be fake.

So who do we think we're fooling?

Answer

- 'Perfectly secret' communication is inherently deniable...

...but most encryption introduces risk of coercion!

- Deniable encryption avoids this side-effect risk.

The purpose is not to 'convince' the coercer, but just to *preempt coercion in the first place*.

Is (Flexible) Deniability Meaningful?

Objection #2

- Wouldn't the coercer request the coins of DenGen & DenEnc?

Is (Flexible) Deniability Meaningful?

Objection #2

- Wouldn't the coercer request the coins of DenGen & DenEnc?

Answer

- He could, but users should just insist they ran Gen & Enc.

Is (Flexible) Deniability Meaningful?

Objection #2

- Wouldn't the coercer request the coins of DenGen & DenEnc?

Answer

- He could, but users should just insist they ran Gen & Enc.

Two cases:

1. Coercer has no further recourse: all's well.

Is (Flexible) Deniability Meaningful?

Objection #2

- Wouldn't the coercer request the coins of DenGen & DenEnc?

Answer

- He could, but users should just insist they ran Gen & Enc.

Two cases:

1. Coercer has no further recourse: all's well.
2. Coercer punishes until he gets what he wants.
   - ⋆ Flexible deniability allows for "crying uncle" (proving true message)

Is (Flexible) Deniability Meaningful?

Objection #2

- Wouldn't the coercer request the coins of DenGen & DenEnc?

Answer

- He could, but users should just insist they ran Gen & Enc.

Two cases:

1. Coercer has no further recourse: all's well.
2. Coercer punishes until he gets what he wants.
   - ⋆ Flexible deniability allows for "crying uncle" (proving true message)
   - ⋆ . . . But so does full deniability! Just use verifiable randomness.

Is (Flexible) Deniability Meaningful?

Objection #2

- Wouldn't the coercer request the coins of DenGen & DenEnc?

Answer

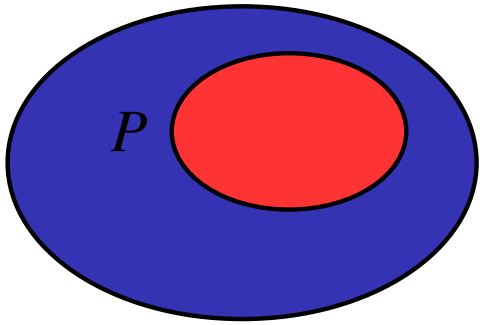
- He could, but users should just insist they ran Gen & Enc.

Two cases:

1. Coercer has no further recourse: all's well.
2. Coercer punishes until he gets what he wants.
   - ⋆ Flexible deniability allows for "crying uncle" (proving true message)
   - ⋆ . . . But so does full deniability! Just use verifiable randomness.
   - ⋆ (Also calls into question the applicability to voting.)

A Tool for Deniability: Translucent Sets [CDNO'97]

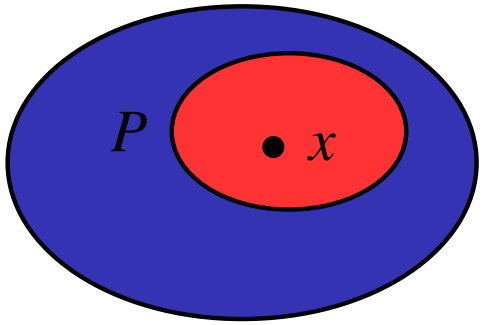
$\{0,1\}^k = U$



Public description pk with secret 'trapdoor' sk.

A Tool for Deniability: Translucent Sets [CDNO'97]

$\{0,1\}^k = U$



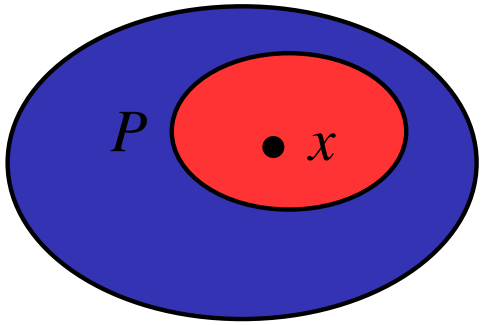
Public description pk with secret 'trapdoor' sk.

Properties

1. Given only pk,
   - ⋆ Can efficiently sample from P (and from U, trivially).
   - ⋆ P-sample is pseudorandom: 'looks like' a U-sample...
   - ⋆ ...so it can be 'faked' as a U-sample.

A Tool for Deniability: Translucent Sets [CDNO'97]

$\{0,1\}^k = U$



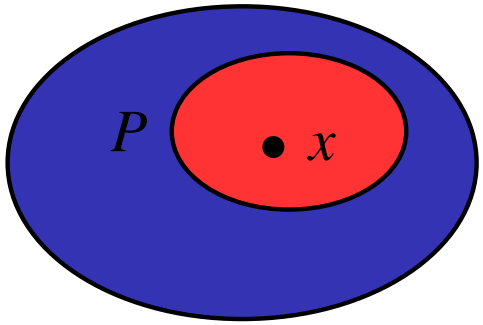
Public description pk with secret 'trapdoor' sk.

Properties

1. Given only pk,
   - ⋆ Can efficiently sample from P (and from U, trivially).
   - ⋆ P-sample is pseudorandom: 'looks like' a U-sample...
   - ⋆ ...so it can be 'faked' as a U-sample.
2. Given sk, can easily distinguish a P-sample from a U-sample.

A Tool for Deniability: Translucent Sets [CDNO'97]

$\{0,1\}^k = U$



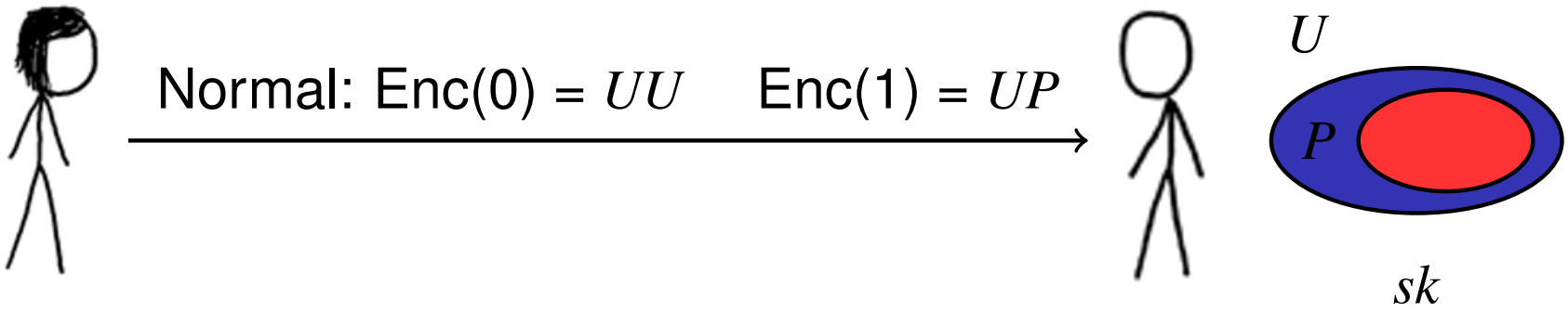
Public description pk with secret 'trapdoor' sk.

Properties

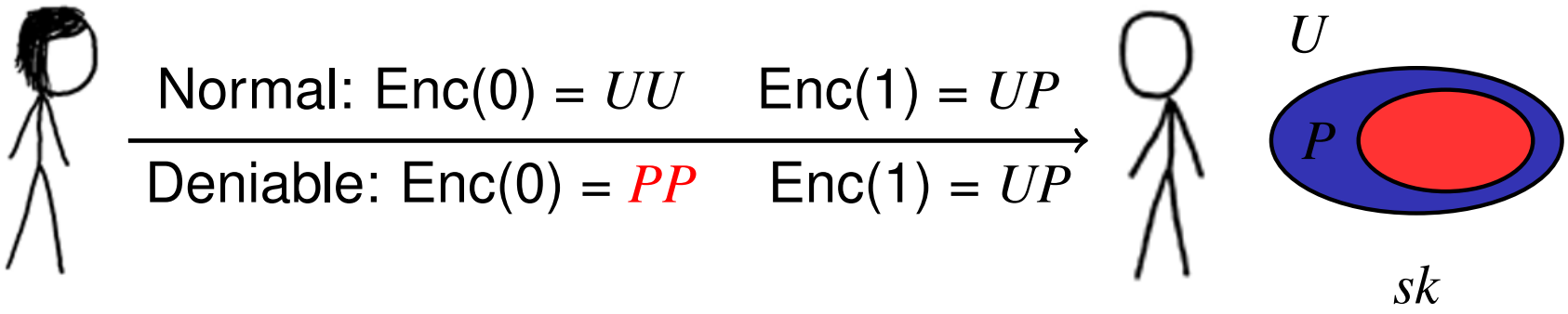
1. Given only pk,
   - ⋆ Can efficiently sample from P (and from U, trivially).
   - ⋆ P-sample is pseudorandom: 'looks like' a U-sample...
   - ⋆ ...so it can be 'faked' as a U-sample.
2. Given sk, can easily distinguish a P-sample from a U-sample.

- Many instantiations: trapdoor perms (RSA), DDH, lattices, ...

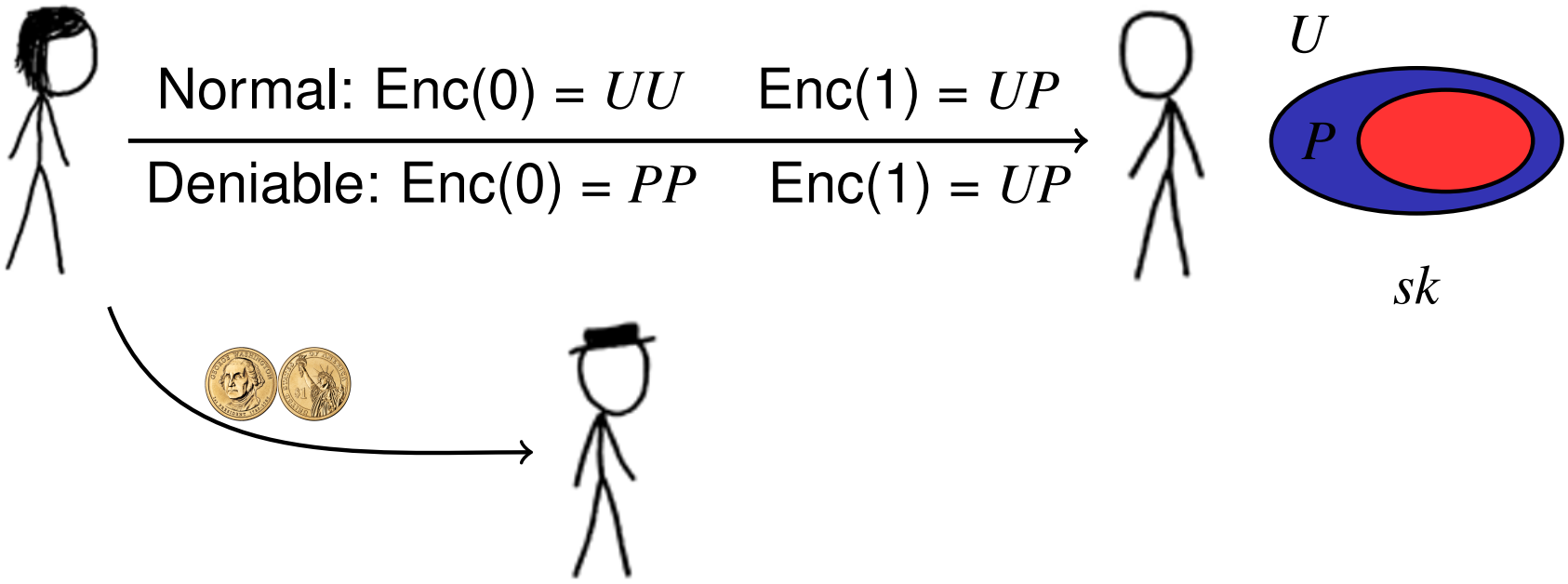
Translucence for Deniability [CDNO'97]



Translucence for Deniability [CDNO'97]



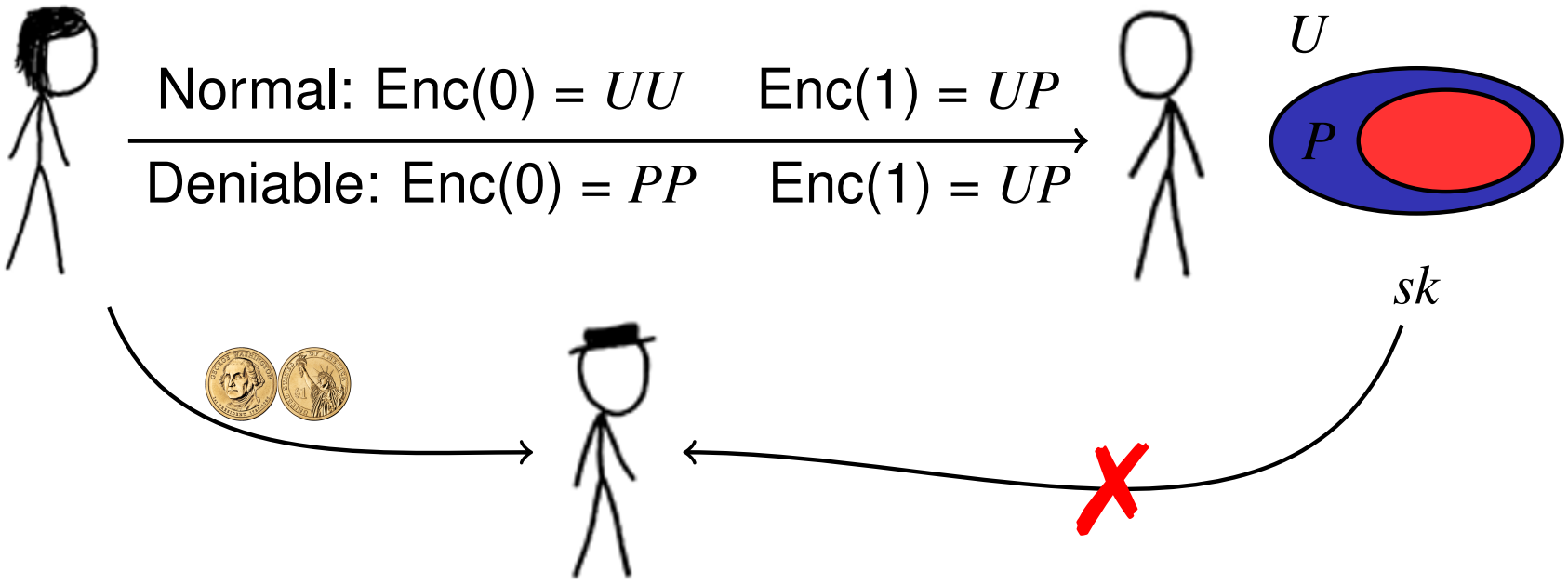
Translucence for Deniability [CDNO'97]



Deniability

- ✔ Alice can fake: $PP \to UP \to UU$

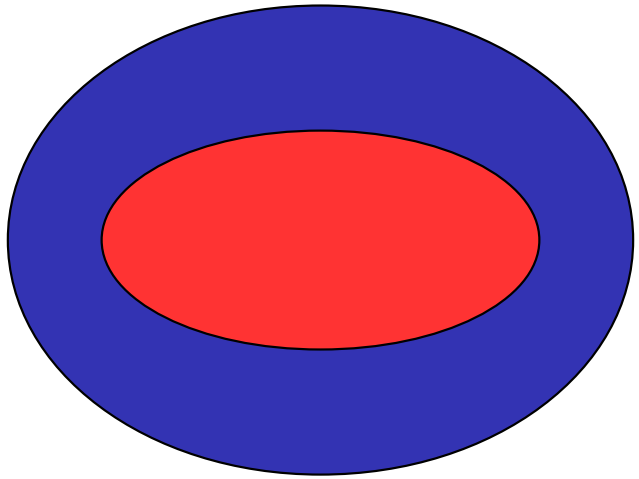
Translucence for Deniability [CDNO'97]



Deniability

- ✔ Alice can fake: $PP \to UP \to UU$
- ✗ What about Bob?? His sk reveals the true message bits!

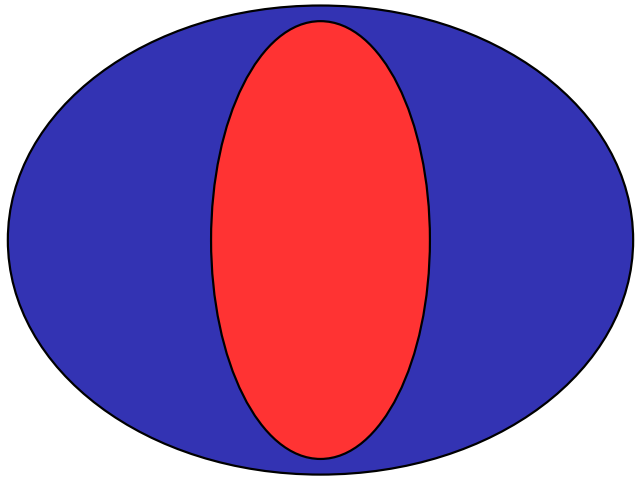
Our Contribution: Bi-Translucent Sets



Properties

1. A pk has many sk, each inducing a *slightly different* P-test.

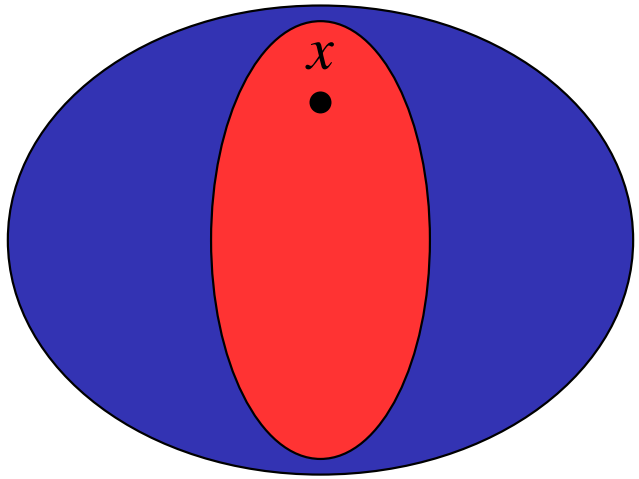
Our Contribution: Bi-Translucent Sets



Properties

1. A pk has many sk, each inducing a *slightly different* P-test.

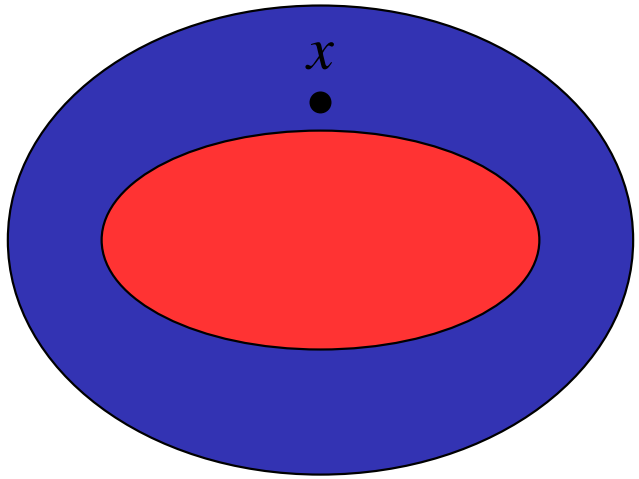
Our Contribution: Bi-Translucent Sets



Properties

1. A pk has many sk, each inducing a *slightly different* P-test.
2. For a given P-sample x, *most* sk classify it correctly.

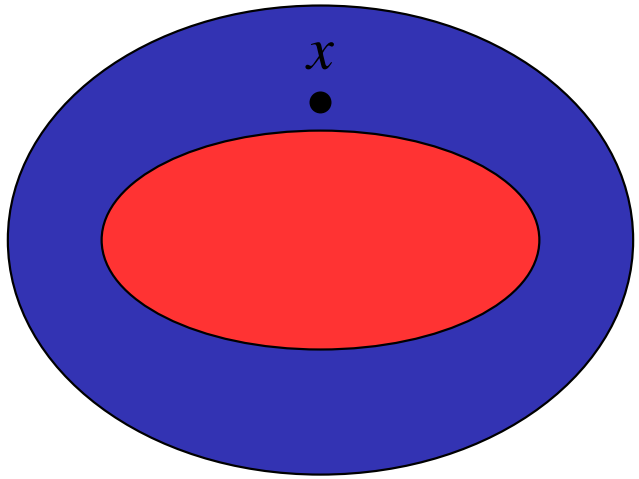
Our Contribution: Bi-Translucent Sets



Properties

1. A pk has many sk, each inducing a *slightly different* P-test.
2. For a given P-sample x, *most* sk classify it correctly.
3. But given a P-sample x and the faking key fk,
   can generate a 'good-looking' sk^* that classifies x as a U-sample.

Our Contribution: Bi-Translucent Sets

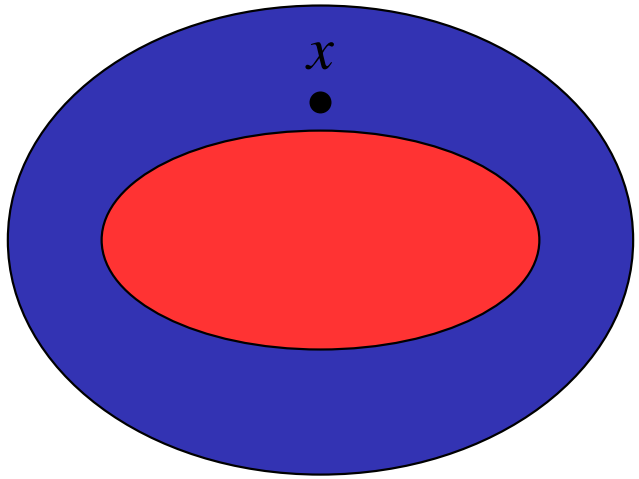


Properties

1. A pk has many sk, each inducing a *slightly different* P-test.
2. For a given P-sample x, *most* sk classify it correctly.
3. But given a P-sample x and the faking key fk,
   can generate a 'good-looking' sk^* that classifies x as a U-sample.

$\Rightarrow$ Bob can also fake $P \rightarrow U$!

Our Contribution: Bi-Translucent Sets



Properties

1. A pk has many sk, each inducing a *slightly different* P-test.
2. For a given P-sample x, *most* sk classify it correctly.
3. But given a P-sample x and the faking key fk,
   can generate a 'good-looking' sk^* that classifies x as a U-sample.

$\Rightarrow$ Bob can also fake $P \rightarrow U$!

$\star\star$ Instantiation idea: in [GPV'08] IBE, authority can induce an "oblivious decryption error" via carefully chosen sk_{id}

Extensions and Open Questions

1. Basic scheme does bit-by-bit encryption to *fresh* public keys.

   (But this is inherent for complete equivocability.)

Extensions and Open Questions

1. Basic scheme does bit-by-bit encryption to *fresh* public keys.

   (But this is inherent for complete equivocability.)

   'Plan-ahead' deniability: encrypt & equivocate a *short* symmetric key that conceals one of 2+ possible *long* messages

Extensions and Open Questions

1. Basic scheme does bit-by-bit encryption to *fresh* public keys.

   (But this is inherent for complete equivocability.)

   'Plan-ahead' deniability: encrypt & equivocate a *short* symmetric key that conceals one of 2+ possible *long* messages

2. Full deniability (unified Gen and Enc), possibly with interaction / trusted setup?

Extensions and Open Questions

1. Basic scheme does bit-by-bit encryption to *fresh* public keys.

   (But this is inherent for complete equivocability.)

   'Plan-ahead' deniability: encrypt & equivocate a *short* symmetric key that conceals one of 2+ possible *long* messages

2. Full deniability (unified Gen and Enc), possibly with interaction / trusted setup?

Thanks!

Full version: ePrint #2011/352