

Automatically Invalidating Game-Based Security Definitions



SKECH

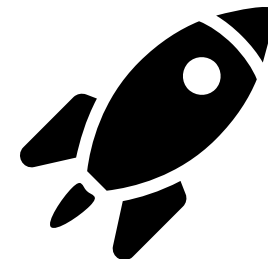
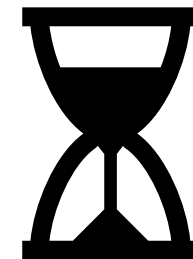
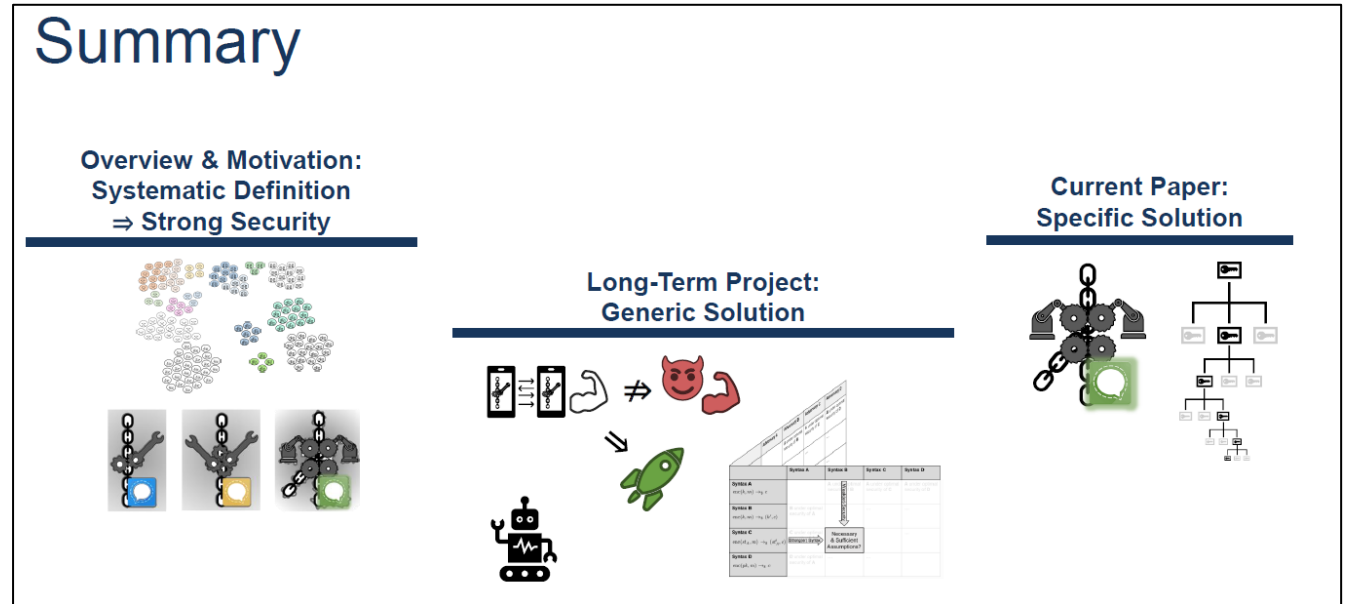
June 24

**Real-World Cryptography Group
FAU Erlangen-Nürnberg, Germany**

Paul Rösler

Disclaimer: Semi-Confidential!

- You're *Early Reviewer 3*
- Workshop
- Discussion
- Ideas
- Work in progress:
 - Extensions
 - Part of larger project



Setting: Game-Based Security Definitions

Game $\text{KIND}_R^b(\mathcal{A})$

```

00 ·  $s_A \leftarrow 0$ ;  $r_B \leftarrow 0$ 
01 ·  $s_B \leftarrow 0$ ;  $r_A \leftarrow 0$ 
02 ·  $e_A \leftarrow 0$ ;  $\text{EP}_A[\cdot] \leftarrow \perp$ 
03 ·  $E_B^+ \leftarrow 0$ ;  $E_B^- \leftarrow 0$ 
04 ·  $\text{adc}_A[\cdot] \leftarrow \perp$ ;  $is_B \leftarrow \text{T}$ 
05 ·  $\text{adc}_B[\cdot] \leftarrow \perp$ ;  $is_A \leftarrow \text{T}$ 
06 ·  $\text{key}_A[\cdot] \leftarrow \perp$ ;  $\text{key}_B[\cdot] \leftarrow \perp$ 
07 ·  $\text{XP}_A \leftarrow \emptyset$ ;  $\text{XP}_B \leftarrow \emptyset$ 
08 ·  $\text{TR}_A \leftarrow \emptyset$ ;  $\text{TR}_B \leftarrow \emptyset$ 
09 ·  $\text{CH}_A \leftarrow \emptyset$ ;  $\text{CH}_B \leftarrow \emptyset$ 
10 ·  $(S_A, S_B) \leftarrow_{\$} \text{init}$ 
11 ·  $b' \leftarrow_{\$} \mathcal{A}$ 
12 · Require  $\text{TR}_A \cap \text{CH}_A = \emptyset$ 
13 · Require  $\text{TR}_B \cap \text{CH}_B = \emptyset$ 
14 · Stop with  $b'$ 

```

Oracle $\text{Snd}_A(ad)$

```

15 · Require  $S_A \neq \perp$ 
16 ·  $(S_A, k, c) \leftarrow_{\$} \text{snd}_A(S_A, ad)$ 
17 · If  $is_A$ :
18 ·  $\text{adc}_A[s_A] \leftarrow (ad, c)$ 
19 ·  $\text{EP}_A[s_A] \leftarrow e_A$ 
20 ·  $\text{key}_A[e_A, s_A] \leftarrow k$ 
21 ·  $s_A \leftarrow s_A + 1$ 
22 · Return  $c$ 

```

Oracle $\text{Rcv}_A(ad, c)$

```

23 · Require  $S_A \neq \perp$ 
24 · If  $is_A \wedge \text{adc}_B[r_A] \neq (ad, c)$ :
25 ·  $is_A \leftarrow \text{F}$ 
26 · If  $r_A \in \text{XP}_B$ :
27 ·  $\text{TR}_A \leftarrow \bigcup \mathbb{N} \times \{s_A, \dots\}$ 
28 · If  $is_A$ :  $e_A \leftarrow e_A + 1$ 
29 ·  $S_A \leftarrow \text{rcv}_A(S_A, ad, c)$ 
30 · If  $S_A = \perp$ : Return  $\perp$ 
31 ·  $r_A \leftarrow r_A + 1$ 
32 · Return

```

Oracle $\text{Rcv}_B(ad, c)$

```

33 · Require  $S_B \neq \perp$ 
34 · If  $is_B \wedge \text{adc}_A[r_B] \neq (ad, c)$ :
35 ·  $is_B \leftarrow \text{F}$ 
36 · If  $r_B \in \text{XP}_A$ :
37 ·  $\text{TR}_B \leftarrow \bigcup \mathbb{N} \times \{r_B, \dots\}$ 
38 · If  $is_B$ :  $E_B^+ \leftarrow \text{EP}_A[r_B]$ 
39 ·  $(S_B, k) \leftarrow \text{rcv}_B(S_B, ad, c)$ 
40 · If  $S_B = \perp$ : Return  $\perp$ 
41 · If  $is_B$ :  $k \leftarrow \diamond$ 
42 ·  $\text{key}_B[E_B^+, r_B] \leftarrow k$ 
43 ·  $r_B \leftarrow r_B + 1$ 
44 · Return

```

Oracle $\text{Snd}_B(ad)$

```

45 · Require  $S_B \neq \perp$ 
46 ·  $(S_B, c) \leftarrow_{\$} \text{snd}_B(S_B, ad)$ 
47 · If  $is_B$ :
48 ·  $\text{adc}_B[s_B] \leftarrow (ad, c)$ 
49 ·  $E_B^- \leftarrow E_B^- + 1$ 
50 ·  $s_B \leftarrow s_B + 1$ 
51 · Return  $c$ 

```

Oracle Expose_A

```

52 · If  $is_A$ :  $\text{XP}_A \leftarrow \bigcup \{s_A\}$ 
53 · Return  $S_A$ 

```

Oracle Expose_B

```

54 ·  $\text{TR}_B \leftarrow \bigcup [E_B^+ \dots E_B^-] \times \{r_B, \dots\}$ 
55 · If  $is_B$ :
56 ·  $\text{XP}_B \leftarrow \bigcup \{s_B\}$ 
57 ·  $\text{TR}_A \leftarrow \bigcup [E_B^+ \dots E_B^-] \times \{r_B, \dots\}$ 
58 · Return  $S_B$ 

```

Oracle $\text{Reveal}(u, i)$

as in URKE (Fig. 5)

Oracle $\text{Challenge}(u, i)$

as in URKE (Fig. 5)

SoK: Game-based Security Models for Group Key Exchange

Bertram Poettering¹, Paul Rösler², Jörg Schwenk³, and Douglas Stebila⁴

¹ IBM Research – Zurich, Rüschlikon, Switzerland

poezurich.ibm.com

² TU Darmstadt, Darmstadt, Germany

paul.roesler@tu-darmstadt.de

³ Ruhr University Bochum, Bochum, Germany

joerg.schwenk@rub.de

⁴ University of Waterloo, Waterloo, Canada

dstebila@uwaterloo.ca

Game $\text{ANON}_{\text{RKE}}^b(\mathcal{A})$

```

00 U-ANONb ← U-ANONRKEb
01 For  $d \in \{0, 1\}$ :
02    $(s_d, r_d) \leftarrow (0, 0)$ 
03    $\text{imp}_d \leftarrow \text{fal}$ 
04    $(\text{stR}, \text{cstR}, \text{cad}) \leftarrow ([\cdot], [\cdot], [\cdot])$ 
05    $\text{stR}[0] \leftarrow \text{U-ANON}_b.\text{stR}$ 
06    $(c, \text{cc}, \text{rcvd}) \leftarrow (\emptyset, \emptyset, \emptyset)$ 
07    $(xS, \text{cxS}) \leftarrow (\emptyset, [\cdot])$ 
08    $(xS, \text{cxS}, xR, \text{cxR}) \leftarrow (\text{fal}, \text{fal}, \text{fal}, \text{fal})$ 
09    $b' \leftarrow_{\$} \mathcal{A}$ 
10   Stop with  $b'$ 

```

Oracle RR

```

11 U-ANONb.RR
12 ·  $(xS, \text{cxS}) \leftarrow (\text{fal}, \text{fal})$ 
13 · Return

```

Oracle Expose_S

```

14 ▷ If  $\text{cxS} = \text{tru}$ : Require  $(s_0, \dots) \notin \text{cxS}$ 
15 ▷ If  $xS = \text{tru} \wedge (s_0, s_1) \notin xS$ :
16 ▷ Require  $(\dots, s_1) \notin xS$ 
17 ◊ If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ :
18 ◊ Require  $\text{cxR} = \text{fal}$ 
19  $\text{stS} \leftarrow \text{U-ANON}_b.\text{Expose}_S$ 
20  $xS \leftarrow \bigcup \{(s_0, s_1)\}$ 
21 ·  $xS \leftarrow \text{tru}$ 
22 · Return  $\text{stS}$ 

```

Oracle Expose_R

```

23 i Require unique = tru
24 ◊ Require  $\text{cxR} = \text{fal}$ 
25 ◊ Require  $\text{imp}_0 = \text{imp}_1$ 
26 If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ :
27 ⊕ For all  $\hat{s} \in \text{cc}$  require  $\hat{s} \leq r_0$ 
28 ◊ Require  $(r_0, \dots) \notin \text{cxS}$ 
29  $\text{stR} \leftarrow \text{U-ANON}_b.\text{Expose}_R$ 
30 ·  $xR \leftarrow \text{tru}$ 
31 · Return  $\text{stR}$ 

```

Oracle ChallExpose_S

```

32 ▷ If  $xS = \text{tru} \vee \text{cxS} = \text{tru}$ :
33 ▷ Require  $(s_0, \dots) \notin \text{cxS} \wedge (s_0, \dots) \notin xS$ 
34 ◊ If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ :
35 ◊ Require  $xR = \text{cxR} = \text{fal}$ 
36  $\text{stS}_b \leftarrow \text{U-ANON}_b.\text{ChallExpose}_S$ 
37 i If  $b = 0$ :  $\text{cstR}.\text{append}(\text{stR}[s_0])$ 
38 i If  $b = 1$ :  $\text{cstR}.\text{append}(\text{U-ANON}_1.\text{ccStR})$ 
39  $\text{cxS}.\text{append}((s_0, s_1))$ 
40 ·  $\text{cxS} \leftarrow \text{tru}$ 
41 · Return  $\text{stS}_b$ 

```

Oracle $\text{Snd}(ad)$

```

42 ⊕ If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ : Require  $\text{cxR} = \text{fal}$ 
43  $(c, k) \leftarrow \text{U-ANON}_b.\text{Snd}(ad)$ 
44  $\text{cad}.\text{append}(c, ad)$ 
45  $c \leftarrow \{s_0\}$ 
46  $s_0 \leftarrow 1$ ,  $s_1 \leftarrow 1$ 
47 i  $(\text{stR}[s_b], \dots) \leftarrow \text{RKE}.\text{rcv}(\text{stR}[s_b - 1], c, ad)$ 
48 · Return  $(c, k)$ 

```

Oracle $\text{ChallSnd}(ad)$

```

49 ⊕ If  $\text{imp}_0 = \text{fal}$ : Require  $xR = \text{cxR} = \text{fal}$ 
50  $(c_b, k_b) \leftarrow \text{U-ANON}_b.\text{ChallSnd}(ad)$ 
51  $\text{cad}.\text{append}(c_b, ad)$ 
52  $\text{cc} \leftarrow \{s_0\}$ 
53  $s_0 \leftarrow 1$ 
54 i If  $b = 0$ :  $(\text{stR}[s_0], \dots) \leftarrow \text{RKE}.\text{rcv}(\text{stR}[s_0 - 1], c_0, ad)$ 
55 · Return  $(c_b, k_b)$ 

```

Oracle $\text{Rcv}(c, ad)$

```

56  $\text{succ}_b \leftarrow \text{U-ANON}_b.\text{Rcv}(c, ad)$ 
57 If  $\exists \hat{r} \geq \min(r_0, r_1)$  s.t.  $(c, ad) = \text{cad}[\hat{r}]$ :
58 If  $b = 0$ :
59  $r'_1 \leftarrow \min(c \setminus \text{rcvd})$ 
60  $\text{succ}_1 \leftarrow \neg \text{imp}_1 \wedge [r'_1 = \hat{r}]$ 
61 If  $\text{succ}_1$ :  $\text{rcvd} \leftarrow \{\hat{r}\}$ 
62 If  $b = 1$ :  $\text{succ}_0 \leftarrow \neg \text{imp}_0 \wedge [r_0 = \hat{r}]$ 
63 If  $\text{succ}_{1-b}$ :  $r_{1-b} \leftarrow 1$ 
64 i Else: //check for impersonations
65 i Let  $\mathcal{S} \subseteq xS$  s.t.  $(\dots, r_1) \in xS$ 
66 i If  $|\mathcal{S}| > 1 \wedge (r_0, \dots) \in \mathcal{S}$ : unique  $\leftarrow \text{fal}$ 
67 i For  $(\hat{r}_0, \hat{r}_1) \in \mathcal{S}$ 
68 i If  $\text{RKE}.\text{rcv}(\text{stR}[\hat{r}_b], c, ad) \neq (\dots, \perp)$ :
69 i  $\text{imp}_0 \leftarrow \text{imp}_0 \vee [r_0 = \hat{r}_0]$ 
70 i  $\text{imp}_1 \leftarrow \text{tru}$ 
71 i If  $\text{imp}_{1-b}$ :  $r_{1-b} \leftarrow 1$ 
72 i  $\mathcal{I} \leftarrow \{i \mid \text{cxS}[i] = (\hat{r}_0, \hat{r}_1) \text{ s.t. } \hat{r}_b = r_b\}$ 
73 i For  $i \in \mathcal{I}$ :
74 i If  $\text{RKE}.\text{rcv}(\text{cstR}[i], c, ad) \neq (\dots, \perp)$ :
75 i  $\text{imp}_0 \leftarrow \text{imp}_0 \vee [r_0 = \hat{r}_0]$ , where  $\hat{r}_0 = \text{cxS}[i][0]$ 
76 i If  $\text{imp}_{1-b}$ :  $r_{1-b} \leftarrow 1$ 
77 If  $\text{succ}_b$ :  $r_b \leftarrow 1$ 
78 · Return

```

Oracle ChallExpose_R

```

79 ◊ Require  $xR = \text{cxR} = \text{fal}$ 
80 ◊ Require  $(r_0, \dots) \notin xS \wedge (r_0, \dots) \notin \text{cxS}$ 
81 ◊ Require  $\text{imp}_0 = \text{fal}$ 
82 ⊕ If  $\text{imp}_1 = \text{fal}$ : Require  $s_0 = r_0$ 
83  $\text{stR}_b \leftarrow \text{U-ANON}_b.\text{ChallExpose}_R$ 
84 ·  $\text{cxR} \leftarrow \text{tru}$ 
85 · Return  $\text{stR}_b$ 

```

Setting: Game-Based Security Definitions

$\text{SE.enc}(k, m) \rightarrow c$

$\text{SE.dec}(k, c) \rightarrow m$

$\text{OW-CCA}_{\text{SE}}(\mathcal{A}) :$

$M[\cdot] \leftarrow \perp; C \leftarrow \emptyset$

$k \leftarrow_{\$} \mathcal{K}$

$(c, m) \leftarrow_{\$} \mathcal{A}$

Stop with $M[c] = m$

$\text{Enc}(m) :$

Return $\text{SE.enc}(k, m)$

$\Pr[\text{SE.dec}(k, \text{SE.enc}(k, m)) = m] = 1$

$\text{Chall}() :$

$m \leftarrow_{\$} \mathcal{M}$

$c \leftarrow \text{SE.enc}(k, m)$

$M[c] \leftarrow m$

Return $C \cup \{c\}$

Return c

$\text{Dec}(c) :$

Return $C \cap \text{SE.dec}(k, c)$

Return $\text{SE.dec}(k, c)$

$\Pr[\text{SE.dec}(k, \text{SE.enc}(k, m)) = m] = 1$

Setting: Game-Based Security Definitions

```

Game KINDRb(A)
00 · sA ← 0; rB ← 0
01 · sB ← 0; rA ← 0
02 · eA ← 0; EPA[·] ← ⊥
03 · EB+ ← 0; EB- ← 0
04 · adcA[·] ← ⊥; isB ← T
05 · adcB[·] ← ⊥; isA ← T
06 · keyA[·] ← ⊥; keyB[·] ← ⊥
07 · XPA ← 0; XPB ← 0
08 · TRA ← 0; TRB ← 0
09 · CHA ← 0; CHB ← 0
10 · (SA, SB) ←s init
11 · b' ←s A
12 · Require TRA ∩ CHA = ∅
13 · Require TRB ∩ CHB = ∅
14 · Stop with b'

Oracle SndA(ad)
15 · Require SA ≠ ⊥
16 · (SA, k, c) ←s sndA(SA, ad)
17 · If isA:
18 ·   adcA[sA] ← (ad, c)
19 ·   EPA[sA] ← eA
20 · keyA[eA, sA] ← k
21 · sA ← sA + 1
22 · Return c

Oracle RcvA(ad, c)
23 · Require SA ≠ ⊥
24 · If isA ∧ adcB[rA] ≠ (ad, c):
25 ·   isA ← F
26 ·   If rA ∈ XPB:
27 ·     TRA ←s N × {sA, ...}
28 · If isA: eA ← eA + 1
29 · SA ← rcvA(SA, ad, c)
30 · If SA = ⊥: Return ⊥
31 · rA ← rA + 1
32 · Return

Oracle RcvB(ad, c)
33 · Require SB ≠ ⊥
34 · If isB ∧ adcA[rB] ≠ (ad, c):
35 ·   isB ← F
36 ·   If rB ∈ XPA:
37 ·     TRB ←s N × {rB, ...}
38 · If isB: EB+ ← EPA[rB]
39 · (SB, k) ← rcvB(SB, ad, c)
40 · If SB = ⊥: Return ⊥
41 · If isB: k ← ∅
42 · keyB[EB+, rB] ← k
43 · rB ← rB + 1
44 · Return

Oracle SndB(ad)
45 · Require SB ≠ ⊥
46 · (SB, c) ←s sndB(SB, ad)
47 · If isB:
48 ·   adcB[sB] ← (ad, c)
49 ·   EB+ ← EB+ + 1
50 · SB ← SB + 1
51 · Return c

Oracle ExposeA
52 · If isA: XPA ←s {sA}
53 · Return SA

Oracle ExposeB
54 · TRB ←s [EB+ .. EB-] × {rB, ...}
55 · If isB:
56 ·   XPB ←s {sB}
57 · TRA ←s [EB+ .. EB-] × {rB, ...}
58 · Return SB

Oracle Reveal(u, i)
as in URKE (Fig. 5)

Oracle Challenge(u, i)
as in URKE (Fig. 5)

```

- List of wrong definitions
 - Missed trivial attacks
 - Validation via proof?!
 - JJ: Proof mistakes
 - Worth of proofs?!
 - Not sure we caught all...
- Use tools!

```

Game ANONRKEb(A)
00 U-ANONb ← U-ANONRKEb
01 For d ∈ {0, 1}:
02   (sd, rd) ← (0, 0)
03   impd ← fal
04 (stR, cstR, cad) ← ([·], [·], [·])
05 stR[0] ← U-ANONb.stR
06 (c, cc, rcvd) ← (∅, ∅, ∅)
07 (xS, cxS) ← (∅, [·])
08 (xS, cxS, xR, cxR) ← (fal, fal, fal, fal)
09 b' ←s A
10 Stop with b'

Oracle RR
11 U-ANONb.RR
12 · (xS, cxS) ← (fal, fal)
13 Return

Oracle ExposeS
14 ▷ If cxS = tru: Require (s0, ...) ∉ cxS
15 ▷ If xS = tru ∧ (s0, s1) ∉ xS:
16 ▷ Require (s0, s1) ∉ xS
17 ◊ If imp0 = imp1 = fal:
18 ◊ Require cxR = fal
19 stS ← U-ANONb.ExposeS
20 xS ←s {(s0, s1)}
21 · xS ← tru
22 Return stS

Oracle ExposeR
23 i Require unique = tru
24 ◊ Require cxR = fal
25 ◊ Require imp0 = imp1
26 If imp0 = imp1 = fal:
27 ⊕ For all s ∈ cc require s ≤ r0
28 ◊ Require (r0, ...) ∉ cxS
29 stR ← U-ANONb.ExposeR
30 · xR ← tru
31 Return stR

Oracle ChallExposeS
32 ▷ If xS = tru ∨ cxS = tru:
33 ▷ Require (s0, ...) ∉ cxS ∧ (s0, ...) ∉ xS
34 ◊ If imp0 = imp1 = fal:
35 ◊ Require xR = cxR = fal
36 stSb ← U-ANONb.ChallExposeS
37 i If b = 0: cstR.append(stR[s0])
38 i If b = 1: cstR.append(U-ANON1.ccStR)
39 cxS.append((s0, s1))
40 · cxS ← tru
41 Return stSb

Oracle Snd(ad)
42 ⊕ If imp0 = imp1 = fal: Require cxR = fal
43 (c, k) ←s U-ANONb.Snd(ad)
44 cad.append(c, ad)
45 c ←s {s0}
46 s0 ← 1, s1 ← 1
47 i (stR[sb], ...) ← RKE.rcv(stR[sb - 1], c, ad)
48 Return (c, k)

Oracle ChallSnd(ad)
49 ⊕ If imp0 = fal: Require xR = cxR = fal
50 (cb, kb) ←s U-ANONb.ChallSnd(ad)
51 cad.append(cb, ad)
52 cc ←s {s0}
53 s0 ← 1
54 i If b = 0: (stR[s0], ...) ← RKE.rcv(stR[s0 - 1], c0, ad)
55 Return (cb, kb)

Oracle Rcv(c, ad)
56 succb ← U-ANONb.Rcv(c, ad)
57 If ∃ r ≥ min(r0, r1) s.t. (c, ad) = cad[r]:
58 If b = 0:
59 r'1 ← min(c \ rcvd)
60 succ1 ← ¬imp1 ∧ [r'1 = r]
61 If succ1: rcvd ← {r}
62 If b = 1: succ0 ← ¬imp0 ∧ [r0 = r]
63 If succ1-b: r1-b ← 1
64 i Else: //check for impersonations
65 i Let S ⊆ xS s.t. (s, r1) ∈ xS
66 i If |S| > 1 ∧ (r0, ...) ∈ S: unique ← fal
67 i For (r̂0, r̂1) ∈ S
68 i If RKE.rcv(stR[r̂b], c, ad) ≠ (s, ⊥):
69 i imp0 ← imp0 ∨ [r0 = r̂0]
70 i imp1 ← tru
71 i If imp1-b: r1-b ← 1
72 i T ← {i | cxS[i] = (r̂0, r̂1) s.t. r̂b = rb}
73 i For i ∈ T:
74 i If RKE.rcv(cstR[i], c, ad) ≠ (s, ⊥):
75 i imp0 ← imp0 ∨ [r0 = r̂0], where r̂0 = cxS[i][0]
76 i If imp1-b: r1-b ← 1
77 If succb: rb ← 1
78 Return

Oracle ChallExposeR
79 ◊ Require xR = cxR = fal
80 ◊ Require (r0, ...) ∉ xS ∧ (r0, ...) ∉ cxS
81 ◊ Require imp0 = fal
82 ⊕ If imp0 = fal: Require s0 = r0
83 stRb ← U-ANONb.ChallExposeR
84 · cxR ← tru
85 Return stRb

```

Finding Trivial Attacks

- Syntax
 - Correctness
 - Adversarial goal
 - Adversarial capabilities
- } \Rightarrow^* Trivial attacks

Attacks against every construction

→ No* algebraic structure

→ Attacks against *ideal* construction

Simplifying Game-Based Definitions

Indistinguishability up to Correctness
and Its Application to Stateful AE

Phillip Rogaway and Yusi Zhang

Computer Science Department
University of California Davis, One Shields Avenue, USA

Simplifying Game-Based Cryptographic Definitions

By

Yusi Zhang

DISSERTATION

Symbolic Perspective on OW-CCA

$$\mathbf{M} \mapsto \mathbf{K} \mid \text{SE.enc}(\mathbf{K}, \mathbf{M})$$

$$m \in M \implies M \vdash m$$

$$M \vdash k, m \implies M \vdash \text{SE.enc}(k, m)$$

$$M \vdash \text{SE.enc}(k, m), k \implies M \vdash m$$

$$\text{OW-CCA}_{\text{SE}}(\mathcal{A}) :$$

$$\frac{}{M[\cdot] \leftarrow \perp}$$

$$\textcircled{k} \leftarrow_{\$} \mathcal{K}$$

$$(\textcircled{k}, \textcircled{m}) \leftarrow_{\$} \mathcal{A}$$

$$\text{Stop with } M[\textcircled{k}] = \textcircled{m}$$

$$\text{Enc}(\textcircled{m}) :$$

$$\frac{}{\text{Return SE.enc}(\textcircled{k}, \textcircled{m})}$$

$$\text{Chall}() :$$

$$\textcircled{m} \leftarrow_{\$} \mathcal{M}$$

$$\textcircled{k} \leftarrow \text{SE.enc}(\textcircled{k}, \textcircled{m})$$

$$M[\textcircled{k}] \leftarrow \textcircled{m}$$

$$C \leftarrow C \cup \{\textcircled{k}\}$$

$$\text{Return } \textcircled{k}$$

$$\text{Dec}(\textcircled{k}, \textcircled{m}) :$$

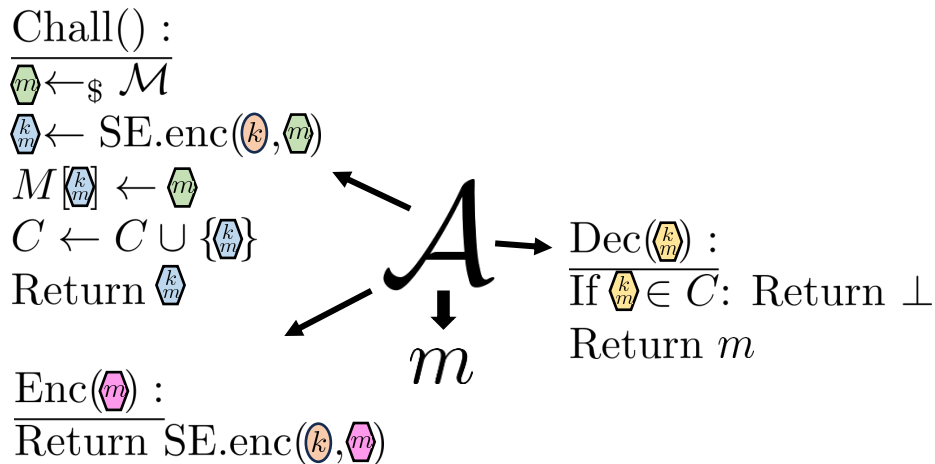
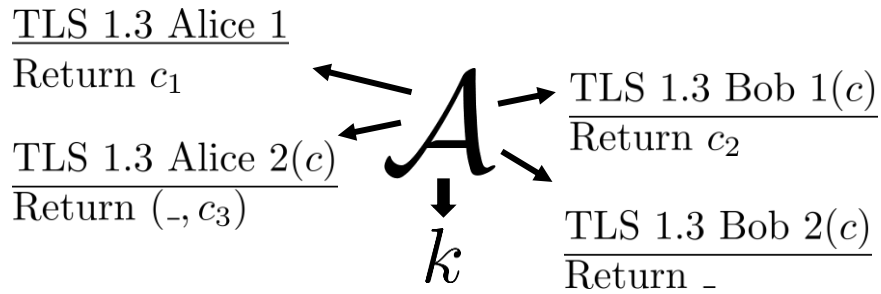
$$\frac{}{\text{If } \textcircled{k} \in C: \text{Return } \perp}$$

$$\text{Return } m$$

Symbolic Verification Tools

SoK: Computer-Aided Cryptography

Manuel Barbosa*, Gilles Barthe^{†‡}, Karthik Bhargavan[§], Bruno Blanchet[§], Cas Cremers[¶], Kevin Liao^{†||}, Bryan Parno^{**}
 *University of Porto (FCUP) and INESC TEC, [†]Max Planck Institute for Security & Privacy, [‡]IMDEA Software Institute, [§]INRIA Paris, [¶]CISPA Helmholtz Center for Information Security, ^{||}MIT, ^{**}Carnegie Mellon University



```

theory symEncOracles
begin
  /* senc and sdec take 2 inputs, resp. */
  functions: senc/2, sdec/2
  /* description of decryption of m is a */
  equations: senc(SeEnc(k,m))=m
  /* ORacles */
  /* Simplest: a fresh challenge key and only permits a single challenge afterwards */
  rule OracleInit:
  [ Fr(-challKey)
  ~ OnlyOnce()
  | OracleInit(-challKey),
  PreChall() ]
  /* Outputs the encryption of input message a under the challenge key */
  rule OracleEnc:
  [ OracleInit(-challKey),
  | In(m) ]
  ->
  [ Out(senc(-challKey,m)) ]
  /* Outputs decryption of input ciphertext under the challenge key as long as the challenge ciphertext has not been created yet */
  /* For this, it requires that the SingleChallenge fact can be consumed and it produces this fact after it consumes it */
  rule OracleDec1:
  [ OracleInit(-challKey),
  PreChall(),
  | In(senc(-challKey,m)) ]
  ->
  [ PreChall(),
  | Out(m) ]
  /* OracleDec2 consumes the SingleChallenge fact and outputs a challenge ciphertext */
  rule OracleDec2:
  [ OracleInit(-challKey),
  PreChall(),
  | Fr(-challMsg) ]
  [ OracleChall(senc(-challKey,-challMsg)),
  | Out(senc(-challKey,-challMsg)) ]
  /* Outputs decryption of input ciphertext under the challenge key after the challenge ciphertext was created, unless the input is the challenge ciphertext */
  /* For this, it requires that the input ciphertext differs from the challenge ciphertext */
  rule OracleDec3:
  [ OracleChall(senc(-challKey,-challMsg)),
  | In(senc(-challKey,m)) ]
  [ ! In(senc(-challKey,-challMsg)),senc(-challKey,m) ] ->
  /* -> */ /* For soundy check, removing limitation of challenge decryption */
  [ Out(m) ]
  /* Specifies the final query of the adversary to which it provides the challenge message; if the input message equals the challenge message, the adversary wins */
  rule OracleFin:
  [ OracleChall(senc(-challKey,-challMsg)),
  | In(m) ]
  [ ! In(-challMsg),
  | AdvWins() ] ->
  [ ]
  restriction Equality:
  "All x y :: SEEnc(x) ==> x = y"
  restriction Inequality:
  "All x :: ! SEEnc(x) ==> F"
  restriction OnlyOnce:
  "All #i #j. OnlyOnce([#i] & OnlyOnce[#j]) ==> #i = #j"
  /* Lemma "it is impossible to obtain -challMsg" */
  lemma oneWaySecurity:
  "not( Ex #i. AdvWins() @i )"
end
  
```

Tool	Unbound	Trace	Equiv	Eq-ty	State	Link
CPSA [▷] [16]	●	●	○	○	●	○
F7 [◊] [17]	●	●	○	●	●	●
↳ F5 [◊] [18]	●	●	○	●	●	●
Maude-NPA [▷] [19]	●	●	● ^d	●	○	○
ProVerif* [†] [20]	●	●	● ^d	●	○	○
↳ fs2pv ^{◊†} [21]	●	●	○	●	○	●
↳ GSVerif* [†] [22]	●	●	○	●	●	○
↳ ProVerif-ATP* [†] [23]	●	●	○	●	○	○
↳ StatVerif* [†] [24]	●	●	● ^d	●	●	○
Scyther [▷] [25]	●	●	○	○	○	○
scyther-proof ^{▷§} [26]	●	●	○	○	○	○
Tamarin* [‡] [27]	●	●	● ^d	●	●	○
↳ SAPIC* [28]	●	●	○	●	●	○
CF-AiSe [▷] [29]	○	●	○	○	●	○
OFMC ^{▷†} [30]	○	●	○	●	○	○
SATMC [▷] [31]	○	●	○	○	●	○
AKISS* [32]	○	○	○	●	●	○
APTE* [33]	○	○	● ^t	○	●	○
DEEPSEC* [34]	○	○	● ^t	○	●	○
SAT-Equiv* [35]	○	○	○	○	○	○
SPEC* [§] [36]	○	○	● ^o	○	○	○

Specification language	Miscellaneous symbols
▷ – security protocol notation	↳ – previous tool extension
* – process calculus	† – abstractions
◊ – multiset rewriting	‡ – interactive mode
◊ – general programming language	§ – independent verifiability

Equational theories (Eq-ty)	Equivalence properties (Equiv)
● – with AC axioms	t – trace equivalence
○ – without AC axioms	o – open bisimilarity
○ – fixed	d – diff equivalence

TABLE I
 OVERVIEW OF TOOLS FOR SYMBOLIC SECURITY ANALYSIS. SEE SECTION II-B FOR MORE DETAILS ON COMPARISON CRITERIA.

Automated Attack Search

- OW-CCA: SKE, PKE, KEM
- SUF-CMA: MAC, DS

- Correctness:
 - Statefulness
 - Loops
 - ...

- Games:
 - ...

```

Game  $\text{KIND}_R^k(\mathcal{A})$ 
00 ·  $s_A \leftarrow 0$ ;  $r_B \leftarrow 0$ 
01 ·  $s_B \leftarrow 0$ ;  $r_A \leftarrow 0$ 
02 ·  $e_A \leftarrow 0$ ;  $\text{EP}_A[\cdot] \leftarrow \perp$ 
03 ·  $E_B^+ \leftarrow 0$ ;  $E_B^- \leftarrow 0$ 
04 ·  $\text{adc}_A[\cdot] \leftarrow \perp$ ;  $is_B \leftarrow \text{T}$ 
05 ·  $\text{adc}_B[\cdot] \leftarrow \perp$ ;  $is_A \leftarrow \text{T}$ 
06 ·  $\text{key}_A[\cdot] \leftarrow \perp$ ;  $\text{key}_B[\cdot] \leftarrow \perp$ 
07 ·  $\text{XP}_A \leftarrow \emptyset$ ;  $\text{XP}_B \leftarrow \emptyset$ 
08 ·  $\text{TR}_A \leftarrow \emptyset$ ;  $\text{TR}_B \leftarrow \emptyset$ 
09 ·  $\text{CH}_A \leftarrow \emptyset$ ;  $\text{CH}_B \leftarrow \emptyset$ 
10 ·  $(S_A, S_B) \leftarrow_s \text{init}$ 
11 ·  $b' \leftarrow_s \mathcal{A}$ 
12 · Require  $\text{TR}_A \cap \text{CH}_A = \emptyset$ 
13 · Require  $\text{TR}_B \cap \text{CH}_B = \emptyset$ 
14 · Stop with  $b'$ 

Oracle  $\text{Snd}_A(ad)$ 
15 · Require  $S_A \neq \perp$ 
16 ·  $(S_A, k, c) \leftarrow_s \text{snd}_A(S_A, ad)$ 
17 · If  $is_A$ :
18 ·    $\text{adc}_A[S_A] \leftarrow (ad, c)$ 
19 ·    $\text{EP}_A[S_A] \leftarrow e_A$ 
20 ·    $\text{key}_A[e_A, S_A] \leftarrow k$ 
21 ·    $s_A \leftarrow s_A + 1$ 
22 · Return  $c$ 

Oracle  $\text{Rcv}_A(ad, c)$ 
23 · Require  $S_A \neq \perp$ 
24 · If  $is_A \wedge \text{adc}_B[r_A] \neq (ad, c)$ :
25 ·    $is_A \leftarrow \text{F}$ 
26 ·   If  $r_A \in \text{XP}_B$ :
27 ·      $\text{TR}_A \xleftarrow{\cup} \mathbb{N} \times \{s_A, \dots\}$ 
28 ·   If  $is_A$ :  $e_A \leftarrow e_A + 1$ 
29 ·    $S_A \leftarrow \text{rcv}_A(S_A, ad, c)$ 
30 · If  $S_A = \perp$ : Return  $\perp$ 
31 ·  $r_A \leftarrow r_A + 1$ 
32 · Return

Oracle  $\text{Rcv}_B(ad, c)$ 
33 · Require  $S_B \neq \perp$ 
34 · If  $is_B \wedge \text{adc}_A[r_B] \neq (ad, c)$ :
35 ·    $is_B \leftarrow \text{F}$ 
36 ·   If  $r_B \in \text{XP}_A$ :
37 ·      $\text{TR}_B \xleftarrow{\cup} \mathbb{N} \times \{r_B, \dots\}$ 
38 ·   If  $is_B$ :  $E_B^+ \leftarrow \text{EP}_A[r_B]$ 
39 ·    $(S_B, k) \leftarrow \text{rcv}_B(S_B, ad, c)$ 
40 · If  $S_B = \perp$ : Return  $\perp$ 
41 · If  $is_B$ :  $k \leftarrow \diamond$ 
42 ·    $\text{key}_B[E_B^+, r_B] \leftarrow k$ 
43 ·    $r_B \leftarrow r_B + 1$ 
44 · Return

Oracle  $\text{Snd}_B(ad)$ 
45 · Require  $S_B \neq \perp$ 
46 ·  $(S_B, c) \leftarrow_s \text{snd}_B(S_B, ad)$ 
47 · If  $is_B$ :
48 ·    $\text{adc}_B[S_B] \leftarrow (ad, c)$ 
49 ·    $E_B^+ \leftarrow E_B^+ + 1$ 
50 ·    $s_B \leftarrow s_B + 1$ 
51 · Return  $c$ 

Oracle  $\text{Expose}_A$ 
52 · If  $is_A$ :  $\text{XP}_A \xleftarrow{\cup} \{s_A\}$ 
53 · Return  $S_A$ 

Oracle  $\text{Expose}_B$ 
54 ·  $\text{TR}_B \xleftarrow{\cup} [E_B^+ .. E_B^-] \times \{r_B, \dots\}$ 
55 · If  $is_B$ :
56 ·    $\text{XP}_B \xleftarrow{\cup} \{s_B\}$ 
57 ·    $\text{TR}_A \xleftarrow{\cup} [E_B^+ .. E_B^-] \times \{r_B, \dots\}$ 
58 · Return  $S_B$ 

Oracle  $\text{Reveal}(u, i)$ 
as in URKE (Fig. 5)

Oracle  $\text{Challenge}(u, i)$ 
as in URKE (Fig. 5)

```

```

Game  $\text{ANON}_{\text{SKE}}^k(\mathcal{A})$ 
00 U-ANON0 ← U-ANON0URKE
01 For  $d \in \{0, 1\}$ :
02    $(s_d, r_d) \leftarrow (0, 0)$ 
03    $\text{imp}_d \leftarrow \text{fal}$ 
04    $(\text{stR}, \text{cstR}, \text{cad}) \leftarrow (\{\cdot\}, \{\cdot\}, \{\cdot\})$ 
05    $\text{stR}[0] \leftarrow \text{U-ANON}_{\text{SKE}}. \text{stR}$ 
06    $(c, \text{cc}, \text{rcnd}) \leftarrow (\emptyset, \emptyset, \emptyset)$ 
07    $(\text{xS}, \text{cxS}) \leftarrow (\emptyset, \{\cdot\})$ 
08    $(\text{xS}, \text{cxS}, \text{xR}, \text{cxR}) \leftarrow (\text{fal}, \text{fal}, \text{fal}, \text{fal})$ 
09    $b' \leftarrow \mathcal{A}$ 
10 Stop with  $b'$ 

Oracle RR
11 U-ANON0.RR
12 ·  $(\text{xS}, \text{cxS}) \leftarrow (\text{fal}, \text{fal})$ 
13 Return

Oracle  $\text{Expose}_g$ 
14 ▷ If  $\text{cxS} = \text{tru}$ : Require  $(s_0, \_) \notin \text{cxS}$ 
15 ▷ If  $\text{xS} = \text{tru} \wedge (s_0, s_1) \notin \text{xS}$ :
16 ▷   Require  $(\_, s_1) \notin \text{xS}$ 
17 ◊ If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ :
18 ◊   Require  $\text{cxR} = \text{fal}$ 
19    $\text{stS} \leftarrow \text{U-ANON}_{\text{SKE}}. \text{Expose}_g$ 
20    $\text{xS} \xleftarrow{\cup} \{(s_0, s_1)\}$ 
21 ·  $\text{xS} \leftarrow \text{tru}$ 
22 Return  $\text{stS}$ 

Oracle  $\text{Expose}_R$ 
23 i Require  $\text{unique} = \text{tru}$ 
24 ◊ Require  $\text{cxR} = \text{fal}$ 
25 ◊ Require  $\text{imp}_0 = \text{imp}_1$ 
26 ◊ If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ :
27 ◊   For all  $s \in \text{cc}$  require  $s \leq r_0$ 
28 ◊   Require  $(r_0, \_) \notin \text{cxS}$ 
29    $\text{stR} \leftarrow \text{U-ANON}_{\text{SKE}}. \text{Expose}_R$ 
30 ·  $\text{xR} \leftarrow \text{tru}$ 
31 Return  $\text{stR}$ 

Oracle  $\text{ChallExpose}_g$ 
32 ▷ If  $\text{xS} = \text{tru} \vee \text{cxS} = \text{tru}$ :
33 ▷   Require  $(s_0, \_) \notin \text{cxS} \wedge (s_0, \_) \notin \text{xS}$ 
34 ◊ If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ :
35 ◊   Require  $\text{xR} = \text{cxR} = \text{fal}$ 
36    $\text{stS}_i \leftarrow \text{U-ANON}_{\text{SKE}}. \text{ChallExpose}_g$ 
37 i If  $b = 0$ :  $\text{cstR}. \text{append}(\text{stR}[s_0])$ 
38 i If  $b = 1$ :  $\text{cstR}. \text{append}(\text{U-ANON}_{\text{SKE}}. \text{ccStR})$ 
39    $\text{cxS}. \text{append}(\{(s_0, s_1)\})$ 
40 ·  $\text{cxS} \leftarrow \text{tru}$ 
41 Return  $\text{stS}_i$ 

Oracle  $\text{Snd}(ad)$ 
42 ◊ If  $\text{imp}_0 = \text{imp}_1 = \text{fal}$ : Require  $\text{cxR} = \text{fal}$ 
43    $(c, k) \xleftarrow{\cup} \text{U-ANON}_{\text{SKE}}. \text{Snd}(ad)$ 
44    $\text{cad}. \text{append}(c, ad)$ 
45    $c \xleftarrow{\cup} \{s_0\}$ 
46    $s_0 \xleftarrow{\cup} 1$ ,  $s_1 \xleftarrow{\cup} 1$ 
47 i  $(\text{stR}[s_0], \_) \leftarrow \text{RKE}. \text{rcv}(\text{stR}[s_0 - 1], c, ad)$ 
48 Return  $(c, k)$ 

Oracle  $\text{ChallSnd}(ad)$ 
49 ◊ If  $\text{imp}_0 = \text{fal}$ : Require  $\text{xR} = \text{cxR} = \text{fal}$ 
50    $(c_0, k_0) \xleftarrow{\cup} \text{U-ANON}_{\text{SKE}}. \text{ChallSnd}(ad)$ 
51    $\text{cad}. \text{append}(c_0, ad)$ 
52    $\text{cc} \xleftarrow{\cup} \{s_0\}$ 
53    $s_0 \xleftarrow{\cup} 1$ 
54 i If  $b = 0$ :  $(\text{stR}[s_0], \_) \leftarrow \text{RKE}. \text{rcv}(\text{stR}[s_0 - 1], c_0, ad)$ 
55 Return  $(c_0, k_0)$ 

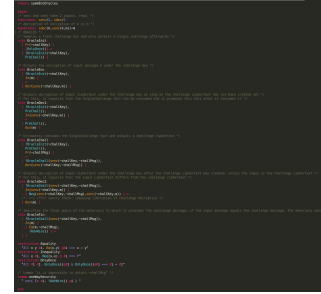
Oracle  $\text{Rcv}(c, ad)$ 
56  $\text{succ}_0 \leftarrow \text{U-ANON}_{\text{SKE}}. \text{Rcv}(c, ad)$ 
57 If  $\exists f \geq \min(r_0, r_1)$  s.t.  $(c, ad) = \text{cad}[f]$ :
58   If  $b = 0$ :
59      $r'_1 \leftarrow \min(c \setminus \text{rcnd})$ 
60      $\text{succ}_1 \leftarrow \sim \text{imp}_1 \wedge [r'_1 = f]$ 
61     If  $\text{succ}_1$ :  $\text{rcnd} \xleftarrow{\cup} \{f\}$ 
62   If  $b = 1$ :  $\text{succ}_0 \leftarrow \sim \text{imp}_0 \wedge [r_0 = f]$ 
63   If  $\text{succ}_{1-b}$ :  $r_{1-b} \xleftarrow{\cup} 1$ 
64 i Else: //check for impersonations
65 i Let  $S \subseteq \text{xS}$  s.t.  $(\_, r_1) \in \text{xS}$ 
66 i If  $|S| > 1 \wedge (r_0, \_) \in S$ :  $\text{unique} \leftarrow \text{fal}$ 
67 i For  $(r_0, r_1) \in S$ :
68 i   If  $\text{RKE}. \text{rcv}(\text{stR}[r_0], c, ad) \neq (\_, \perp)$ :
69 i      $\text{imp}_0 \leftarrow \text{imp}_0 \vee [r_0 = r_0]$ 
70 i      $\text{imp}_1 \leftarrow \text{tru}$ 
71 i   If  $\text{imp}_{1-b}$ :  $r_{1-b} \xleftarrow{\cup} 1$ 
72 i    $Z \leftarrow \{i \mid \text{cxS}[i] = (r_0, r_1) \text{ s.t. } r_b = r_b\}$ 
73 i   For  $i \in Z$ :
74 i     If  $\text{RKE}. \text{rcv}(\text{cstR}[i], c, ad) \neq (\_, \perp)$ :
75 i        $\text{imp}_0 \leftarrow \text{imp}_0 \vee [r_0 = r_0]$ , where  $r_0 = \text{cxS}[i][0]$ 
76 i       If  $\text{imp}_{1-b}$ :  $r_{1-b} \xleftarrow{\cup} 1$ 
77 If  $\text{succ}_b$ :  $r_b \xleftarrow{\cup} 1$ 
78 Return

Oracle  $\text{ChallExpose}_R$ 
79 ◊ Require  $\text{xR} = \text{cxR} = \text{fal}$ 
80 ◊ Require  $(r_0, \_) \notin \text{xS} \wedge (r_0, \_) \notin \text{cxS}$ 
81 ◊ Require  $\text{imp}_0 = \text{fal}$ 
82 ◊ If  $\text{imp}_1 = \text{fal}$ : Require  $s_0 = r_0$ 
83    $\text{stR}_b \leftarrow \text{U-ANON}_{\text{SKE}}. \text{ChallExpose}_R$ 
84 ·  $\text{cxR} \leftarrow \text{tru}$ 
85 Return  $\text{stR}_b$ 

```

Actual Idea Direction

- Implement definitions
- Verify definitions / Reproduce attacks
- Prove: Symbolic attacks \Rightarrow Trivial attacks
- Prove: No symbolic attacks \Rightarrow No trivial attacks



- WIP:
 - Feedback
 - Discussion
 - Happy to collaborate
- Q:
 - Which tools?
 - Discussing proofs
 - Simple fail examples

```

Game ANONSE(A)
00 U-ANONSE ← U-ANONSE
01 For d ∈ {0,1}
02 (s_A, r_A) ← (0,0)
03 (s_B, r_B) ← (0,0)
04 (k, m) ← (0,0)
05 (c, s) ← (0,0)
06 (s_A, r_A) ← (0,0)
07 (s_B, r_B) ← (0,0)
08 (k, m) ← (0,0)
09 (c, s) ← (0,0)
10 Stop with #

Oracle RevB(ad, c)
33 Require S_B ≠ ⊥
34 If is_B ∧ ad_A[r_B] ≠ (ad, c):
35   is_B ← F
36 If r_B ∈ XP_A:
37   TR_B  $\stackrel{d}{\leftarrow}$  N × {r_B, ...}
38 If is_B: E_B' ← EP_A[r_B]
39 (S_B, k) ← rcv_B(S_B, ad, c)
40 If S_B = ⊥: Return ⊥
41 If is_B: k ← 0
42 key_B[E_B', r_B] ← k
43 r_B ← r_B + 1
44 Return

Oracle SndB(ad)
45 Require S_B ≠ ⊥
46 (S_B, c) ← s_snd_B(S_B, ad)
47 If is_B:
48   ad_cp[s_B] ← (ad, c)
49   E_B' ← E_B + 1
50 s_B ← s_B + 1
51 Return c

Oracle ExposeA
52 If is_A: XP_A  $\stackrel{d}{\leftarrow}$  {s_A}
53 Return S_A

Oracle RevA(ad, c)
54 TR_B  $\stackrel{d}{\leftarrow}$  [E_B', E_B] × {r_B, ...}
55 If is_B:
56   XP_B  $\stackrel{d}{\leftarrow}$  {s_B}
57 TR_A  $\stackrel{d}{\leftarrow}$  [E_A', E_A] × {r_A, ...}
58 Return S_B

Oracle Reveal(u, i)
as in URKE (Fig. 5)

Oracle Challenge(u, i)
as in URKE (Fig. 5)
    
```

Tool	Unbound	Trace	Equiv	Eq-thy	State	Link
CPSA ² [16]	•	•	•	•	•	•
FPS ² [17]	•	•	•	•	•	•
4FS ² [18]	•	•	•	•	•	•
Mauds-NPA ² [19]	•	•	•	•	•	•
ProVerif [†] [20]	•	•	•	•	•	•
4fs2pv ² [21]	•	•	•	•	•	•
4GSVerif [†] [22]	•	•	•	•	•	•
4ProVerif-ATP [†] [23]	•	•	•	•	•	•
4StatVerif [†] [24]	•	•	•	•	•	•
Scyther ² [25]	•	•	•	•	•	•
scyster-proof ² [26]	•	•	•	•	•	•
Tamarin [†] [27]	•	•	•	•	•	•
4SAPIC ² [28]	•	•	•	•	•	•
CF-AISE ² [29]	•	•	•	•	•	•
OFMC ² [30]	•	•	•	•	•	•
SATMC ² [31]	•	•	•	•	•	•
AKISS ² [32]	•	•	•	•	•	•
APTE ² [33]	•	•	•	•	•	•
DEEPESEC ² [34]	•	•	•	•	•	•
SAT-Equiv ² [35]	•	•	•	•	•	•
SPEC ² [36]	•	•	•	•	•	•

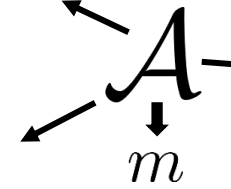
Specification language: ▷ - security protocol notation, + - process calculus, * - multiset rewriting, ○ - general programming language. Miscellaneous symbols: † - previous tool extension, ‡ - abstractions, † - interactive mode, § - independent verifiability. Equational theories (Eq-thy): ● - with AC axioms, ○ - without AC axioms, ○ - fixed. Equivalence properties (Equiv): † - trace equivalence, ○ - open bisimilarity, d - diff equivalence.

OW-CCA_{SE}(\mathcal{A}):
 $M[\cdot] \leftarrow \perp$
 $k \leftarrow_{\$} \mathcal{K}$
 $(\frac{k}{m}) \leftarrow_{\$} \mathcal{A}$
 Stop with $M[\frac{k}{m}] = \frac{m}{m}$

Enc($\frac{m}{m}$):
 Return SE.enc($k, \frac{m}{m}$)

Chall():
 $\frac{m}{m} \leftarrow_{\$} \mathcal{M}$

$\frac{k}{m} \leftarrow$ SE.enc($k, \frac{m}{m}$)
 $M[\frac{k}{m}] \leftarrow \frac{m}{m}$
 $C \leftarrow C \cup \{\frac{k}{m}\}$
 Return $\frac{k}{m}$



Chall():
 $\frac{m}{m} \leftarrow_{\$} \mathcal{M}$
 $\frac{k}{m} \leftarrow$ SE.enc($k, \frac{m}{m}$)
 $M[\frac{k}{m}] \leftarrow \frac{m}{m}$
 $C \leftarrow C \cup \{\frac{k}{m}\}$
 Return $\frac{k}{m}$

Dec($\frac{k}{m}$):
 If $\frac{k}{m} \in C$: Return \perp
 Return m

Dec($\frac{k}{m}$):
 If $\frac{k}{m} \in C$: Return \perp
 Return m

Enc($\frac{m}{m}$):
 Return SE.enc($k, \frac{m}{m}$)