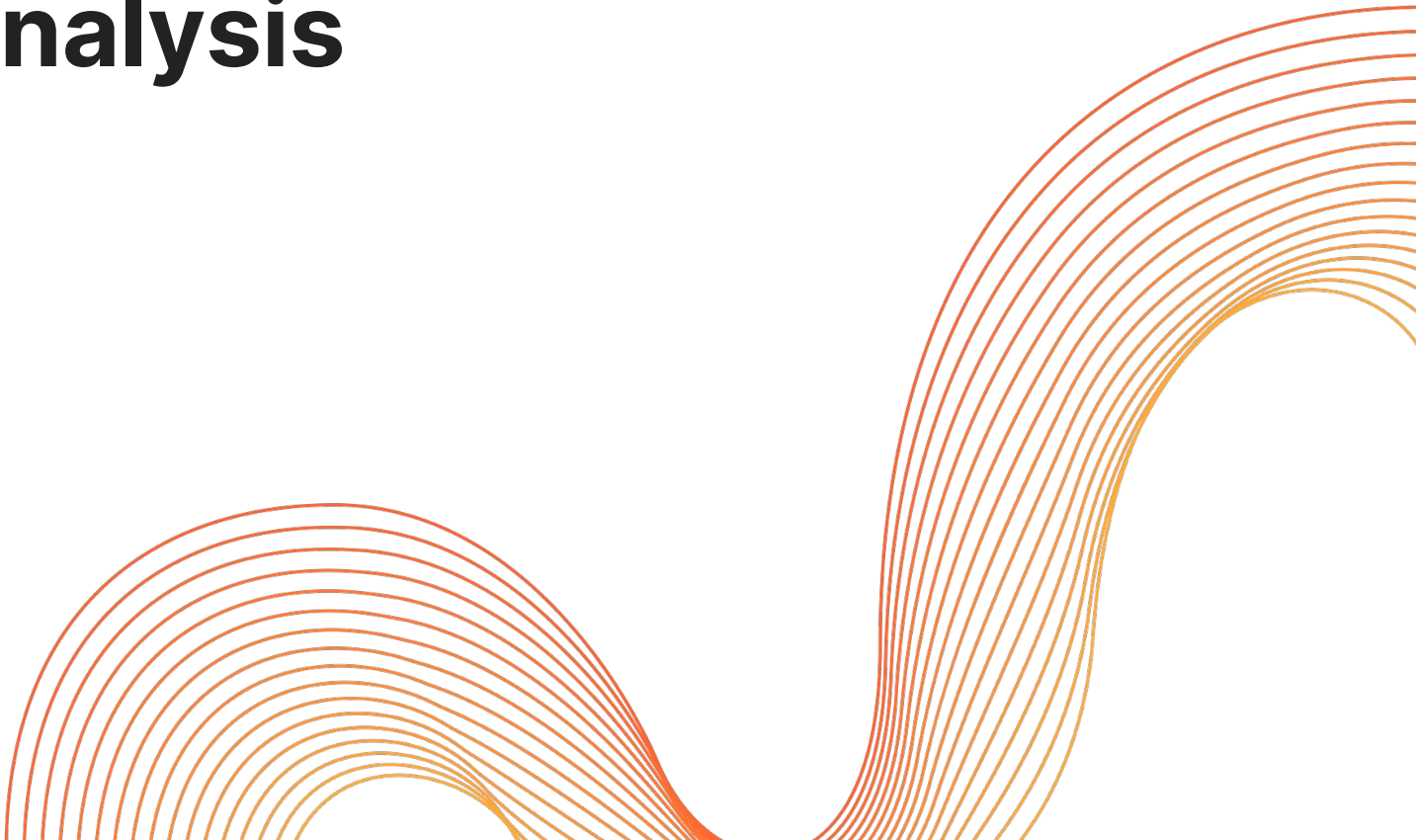
CLOUDFLARE

Formal Analysis

A Brief Introduction



What is Formal Analysis?

Prove that a protocol specification meets its goals

Symbolic vs Computational Analysis

Symbolic analysis:

- Represent protocol algebraically
- Assume cryptographic primitives are perfect
- Prove the protocol meets / doesn't meet its (intended) goals

Computational analysis:

- Represent protocol algebraically
- Use concrete bounds on cryptographic primitives
- Compute an exact security bound for the protocol

CLOUDFLARE

Case Study: TLS 1.3

- Cremers et al. produced a Tamarin model of various drafts of TLS 1.3
- Symbolic analysis
- Highly detailed model capturing virtually all modes and features
- In a single model
- During the standardisation process
- Found and fixed bugs in the design
- Proof very large (> 750k steps)
- Took several days on a 500GB - 128 core server

Needham-Schroeder

$$A \rightarrow B : \{N_A, A\}_{PK_B}$$

$$B \rightarrow A : \{N_A, N_B\}_{PK_A}$$

$$A \rightarrow B : \{N_B\}_{PK_B}$$

CLOUDFLARE

Attack

$$A \rightarrow I : \{N_A, A\}_{PK_I}$$

$$I_A \rightarrow B : \{N_A, A\}_{PK_B}$$

$$B \rightarrow I_A : \{N_A, N_B\}_{PK_A}$$

$$I \rightarrow A : \{N_A, N_B\}_{PK_A}$$

$$A \rightarrow I : \{N_B\}_{PK_I}$$

$$I_A \rightarrow B : \{N_B\}_{PK_B}$$

Needham-Schroeder-Lowe

$$A \rightarrow B : \{N_A, A\}_{PK_B}$$
$$B \rightarrow A : \{N_A, N_B, B\}_{PK_A}$$
$$A \rightarrow B : \{N_B\}_{PK_B}$$

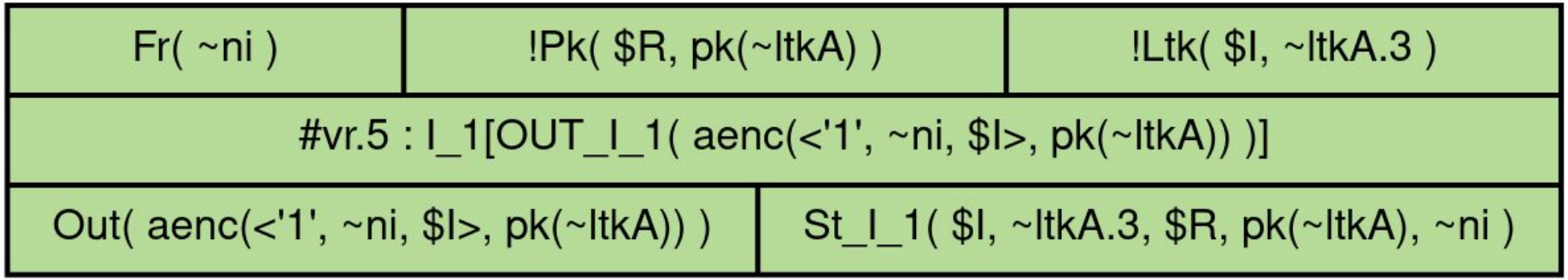
Tamarin

```
rule I_1:
  let m1 = aenc{'1', ~ni, $I}pkR
  in
   [ Fr(~ni)
   , !Pk($R, pkR)
   , !Ltk($I, ltkI)
   ]
  --[ OUT_I_1(m1)
   ]->
   [ Out( m1 )
   , St_I_1($I, ltkI, $R, pkR, ~ni)
   ]
```

Cribbed from the examples distributed with Tamarin

CLOUDFLARE

Tamarin UI



```
lemma nonce_secrecy:
  " /* It cannot be that */
    not(
        Ex A B s #i.
        /* somebody claims to have setup a shared secret, */
        Secret(A, B, s) @ i
        /* but the adversary knows it */
        & (Ex #j. K(s) @ j)
        /* without having performed a long-term key reveal. */
        & not (Ex #r. RevLtk(A) @ r)
        & not (Ex #r. RevLtk(B) @ r)
    )"
```

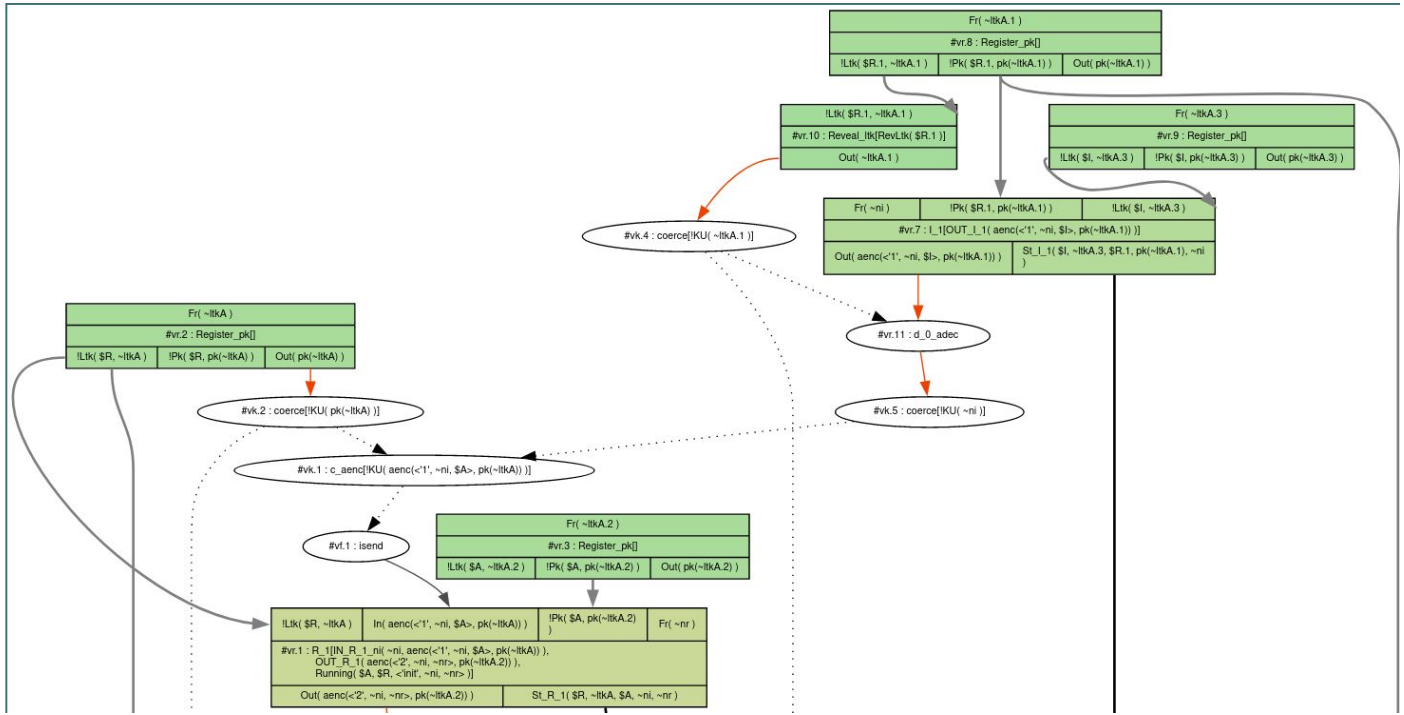
Proof scripts

```
lemma nonce_secrecy:
  all-traces
  "¬(∃ A B s #i.
          (((Secret( A, B, s ) @ #i) ∧ (∃ #j.
K( s ) @ #j)) ∧
           (¬(∃ #r. RevLtk( A ) @ #r))) ∧
          (¬(∃ #r. RevLtk( B ) @ #r)))"
simplify
solve( Secret( A, B, s ) ▶₀ #i )
  case I_2_case_1
  by sorry
next
  case I_2_case_2
  by sorry
next
  case R_2_case_1
  solve( !KU( aenc(<'3', ~nr>, pk(~ltkA)) ) @
#vk.1 )
    case I_2
    by sorry
  next
    case c_aenc
    solve( !KU( aenc(<'1', ni, $A>, pk(~ltkA))
           ) @ #vk.3 )
      case I_1
      by sorry
    next
      case c_aenc
      solve( !KU( pk(~ltkA) ) @ #vk.4 )
        case Register_pk
        solve( !KU( ~nr ) @ #vk.3 )
          case I_2
          solve( !KU( ~ltkA.2 ) @ #vk.12 )
            case Reveal_ltk
            solve( !KU( ~ni ) @ #vk.10 )
              case I_1
              solve( !KU( aenc(<'2', ~ni, ~nr>,
pk(~ltkA.2))
                     ) @ #vk.12 )
                case R_1
                SOLVED // trace found
              qed
            qed
          qed
        qed
      next
        case c_pk
        by sorry
      qed
    qed
  qed
next
  case R_2_case_2
  by sorry
qed
```

Visualization display

Constraint System is Solved

Constraint system



Conclusions

Formal Analysis:

- can be used to prove that a protocol works as intended
- can be very difficult
- has tooling that mechanises a lot of the drudgery away
- has been used to find and fix bugs in protocols we care about

Questions?