Security & Privacy in 3G/4G/5G networks: The AKA Protocol

With S.Alt, P.-A. Fouque, G. Macario-Rat, B. Richard

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

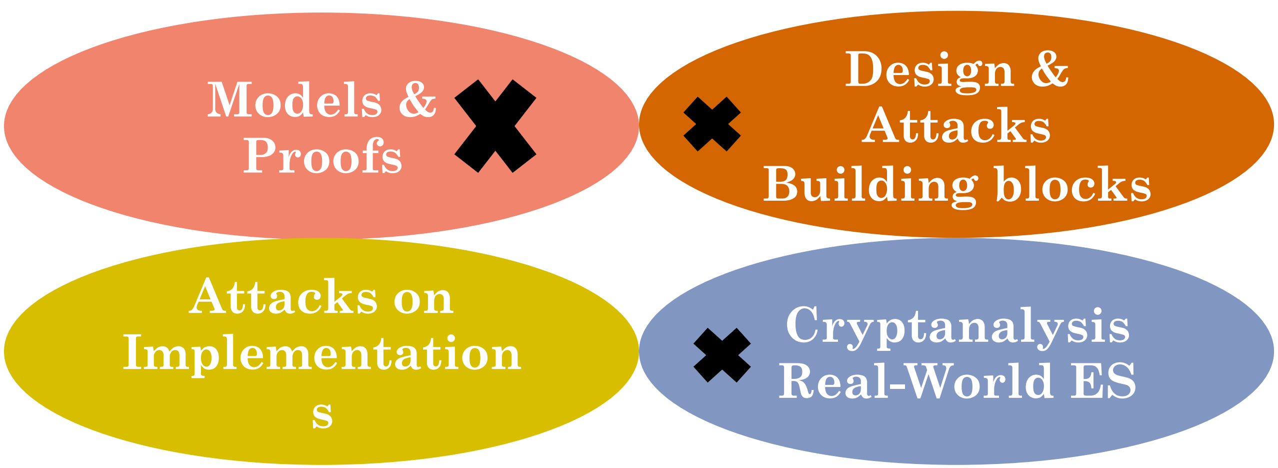
UMR IRISA

Me, myself, and EMSEC

- BSc. & MSc. Mathematics, TU Eindhoven
  - Master thesis on multiparty pairing-based key exchange (supervisors: Tanja Lange, Bernhard Esslinger)
- Ph.D. at CASED (Darmstadt)
  - Thesis: “Security aspects of Distance-Bounding Protocols” (supervisor: Marc Fischlin)
- Post-doc at IRISA (Rennes)
  - ’13-’14: Privacy & Distance Bounding (CIDRE)
  - ‘14-’15: Privacy in geolocation (CIDRE/CAPPRIS)
  - ’15-’16: TLS/SSL (EMSEC)

EMSEC

- IRISA research team
  - Founded 2014
  - Led by: Pierre-Alain Fouque (UR1) & Gildas Avoine (INSA)
  - As of Sept. 2015: 5 permanents: 2 UR1, 2 INSA, 1 CNRS
- Topics: Embedded Security and Cryptography



What I Do

- Distance-Bounding Protocols
  - Security framework [DFKO11, FO12, FO13b],
  - Protocol assessment/comparison [FO13a, MOV14]
  - Privacy-preserving DB [HPO13,GOR14, MOV14]
  - Protocol with Secret Sharing [GKL+14]
  - Implementations [GLO15]

- Authenticated Key-Exchange
  - OPACITY [DFG+13]
  - TLS 1.3 [KMO+15], TLS 1.2&1.3 – ePrint version
  - AKA [AFM+15, FMO+15] (submissions)

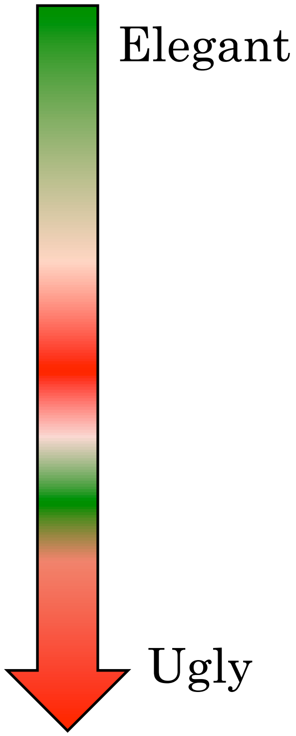
What I Do (II)

- Other primitives
  - Signatures of knowledge [FO11]
  - Redactable signatures for tree data [BBD+10]
  - Anonymous PKE [KMO+13]
  - Private asymmetric fingerprinting [FGLO14]

- Projects
  - ANR LYRICS [finished mid '14]
  - CAPPRIS (Inria) [ongoing]

This Talk

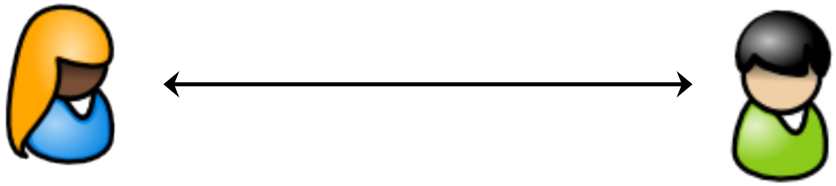
- Authenticated Key Exchange
  - Unilateral/Mutual Authentication
  - Desired Properties
  - Privacy in Authentication
- The AKA Protocol
  - Description
  - Security (intuition)
- AKA and Privacy
  - The case of the Hopeless Task



Part II
Authenticated Key Exchange

Authenticated Key-Exchange

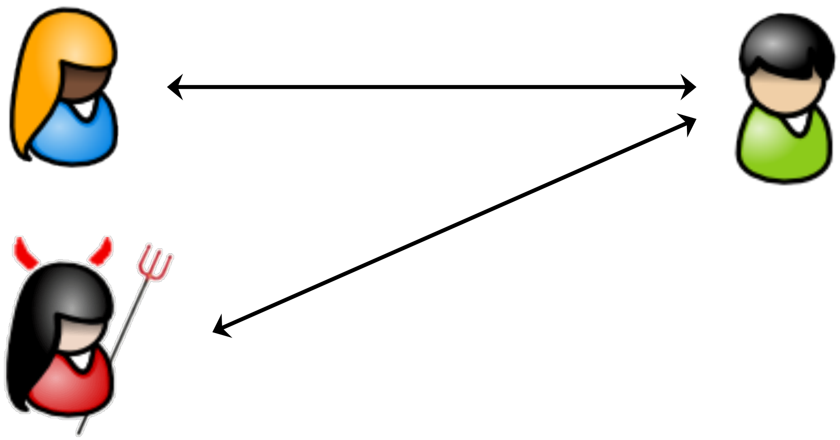
- Allows two parties to communicate securely
  - Peer-to-Peer or Server-Client
  - Examples: TLS/SSL (https://)
- Two steps:
  - Compute session-specific keys (handshake)
  - Use keys for secure communication (symmetric AE)



AKE with Unilateral Authentication

- Usually the case for Server-Client AKE
  - "Anybody" can talk to the server
  - Most common TLS mode

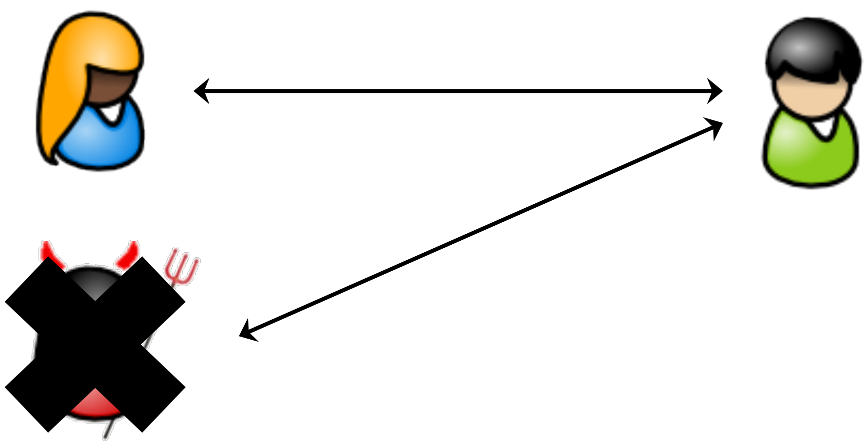
Secure channel server/client or adv/server



AKE with Mutual Authentication

- Sometimes server-client, mostly peer-to-peer
  - Can also be achieved by unilateral authentication + password-based authentication in secure channel [KMO +15]

Client and server confirm partner's identities



AKE Security Properties (Unilateral)

- Key Secrecy [BR93], [BPR00], [CK01]…:
  - **Adversary's goal**: distinguishing the keys of an honest, fresh session from random keys of same length
  - **Rules of game**: adaptive party corruptions, key-reveal, concurrent sessions and interactions

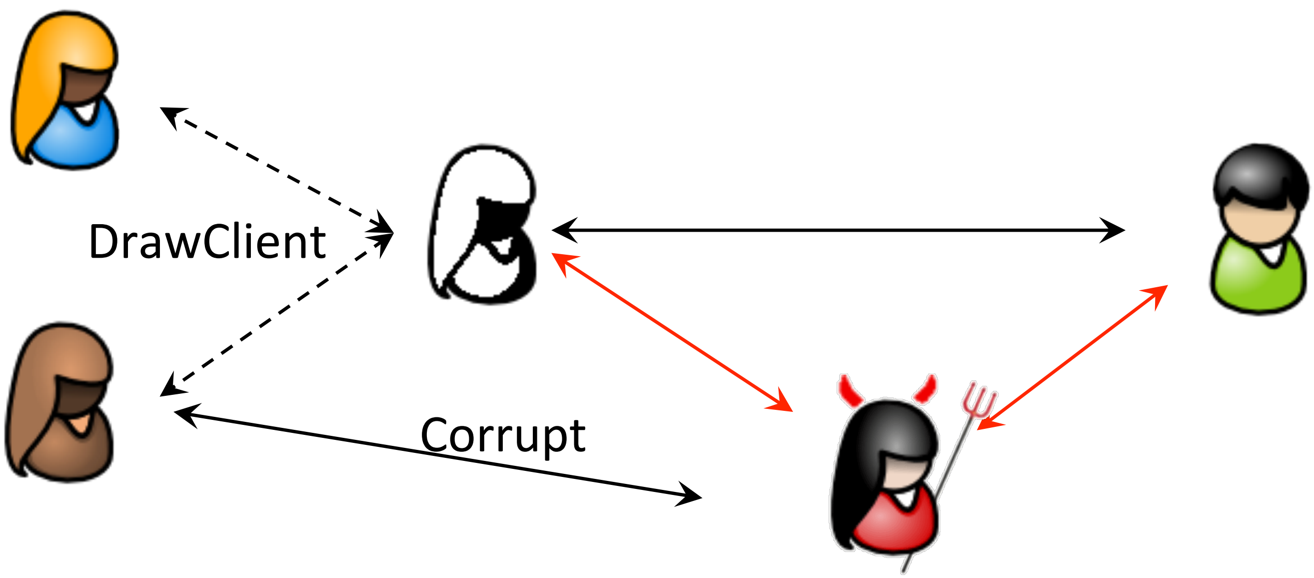
Symmetric Key Restriction: no terminal corruptions!

- Client-impersonation resistance
  - **Adversary's goal**: impersonate client in fresh authentication session
  - **Rules of game**: adaptive party corruptions, key-reveal, concurrent sessions and interactions, no relays!

Terminal Impersonation

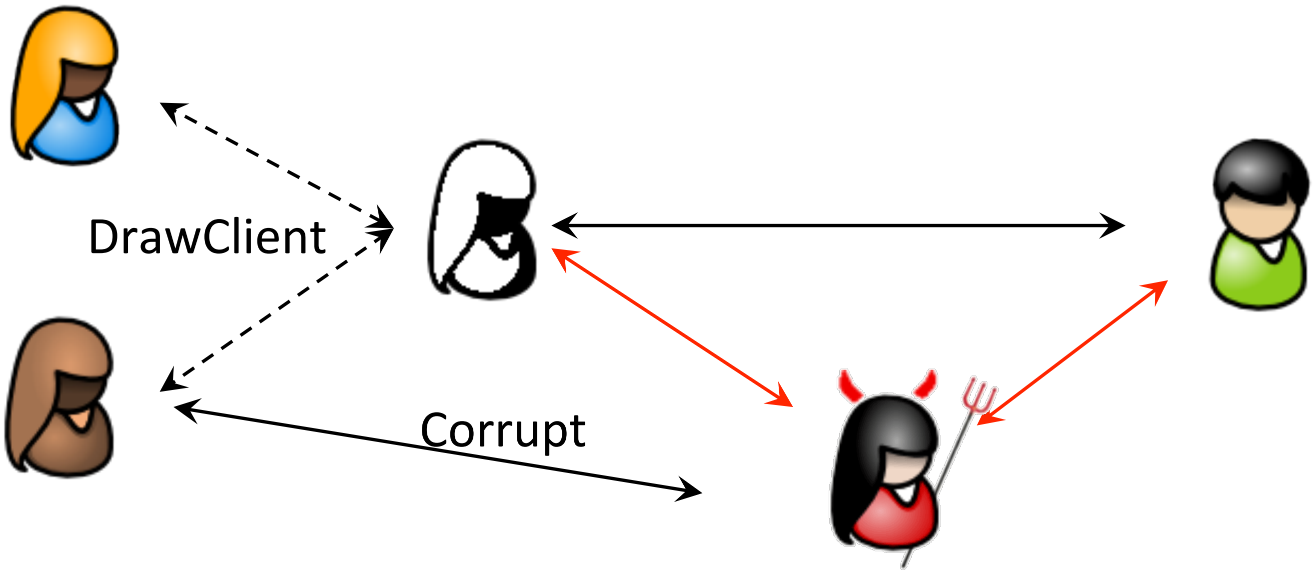
- Terminal-impersonation resistance
  - **Adversary's goal**: impersonate terminal in fresh authentication session
  - **Rules of game**: adaptive party corruptions, key-reveal, concurrent sessions and interactions, no relays!
- The eternal debate: first or second
  - Should terminal authenticate first or second?
  - VANET, MANET, RFID authentication: terminal first
  - When optional, usually terminal second

Privacy in Authentication



- Key Secrecy [JW00], [Vau07], [HPV+12]…:
  - **Adversary's goal**: find input bit to DrawProver
  - **Rules of game**: DrawClient always takes same input bit, can corrupt*, interact, etc.

Privacy Notions



- **Weak** : no corruptions
- **Forward** : once A corrupts, only corruptions (find past LoR connection)
- **Strong** : no restrictions
- Narrow/wide: know result of honest sessions

Impossibility Results

[Vau07]: Strong Privacy requires Key-Agreement

[PV08]: Wide-Forward privacy with symmetric keys is impossible if all state is revealed*

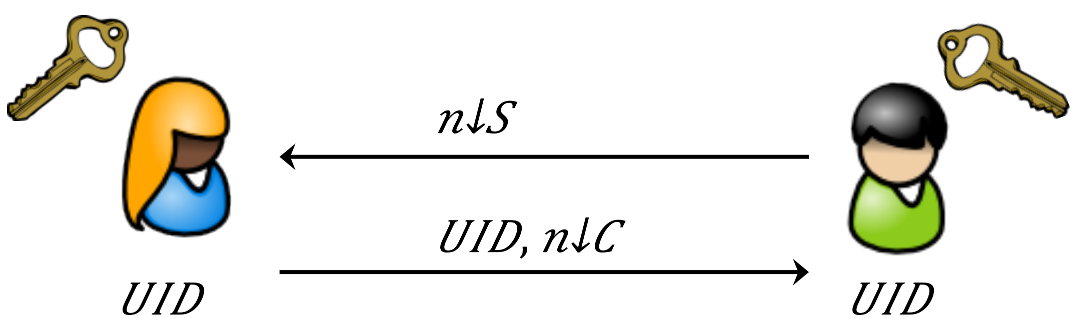
Part III
The AKA Protocol

Part III. 1
Identifiers & Secrets

Elegant Symmetric (A)KE [BR93]

- Usual case for AKE: 2 parties, e.g. client/server
- Share symmetric secret key *sk*
- Sometimes public identifier *UID*
- Elegant KE: use PRF keyed with *sk*

AKE? No problem, use another PRF and switch!



$$Keys := PRF_{sk}(n_C, n_S)$$

The case of 3G/4G/5G

- Usual case for AKE: 2 parties, e.g. client/server
- In 3G/4G/5G networks, 3 parties:
  - Client: registering with (only one) operator

    client key and operator key stored* in SIM
  - Operator: has list of clients, whose data he knows
  - Local terminal: not always operator (think of roaming)

    can authenticate/communicate with client

    must not know keys

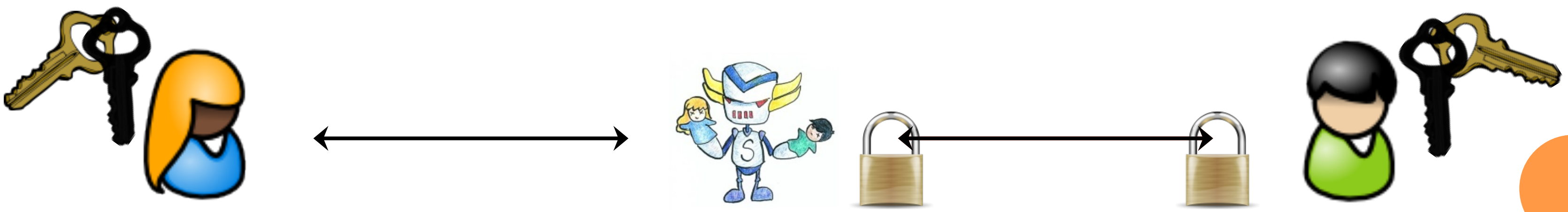
THE CASE OF 3G/4G/5G (CONTD.)

- Some more restrictions:
  - Connection Terminal – Operator is expensive!

    Assumed to take place on secure channel

  - Whenever PKI is used, in practice this means storing PKs and certificates in the phone

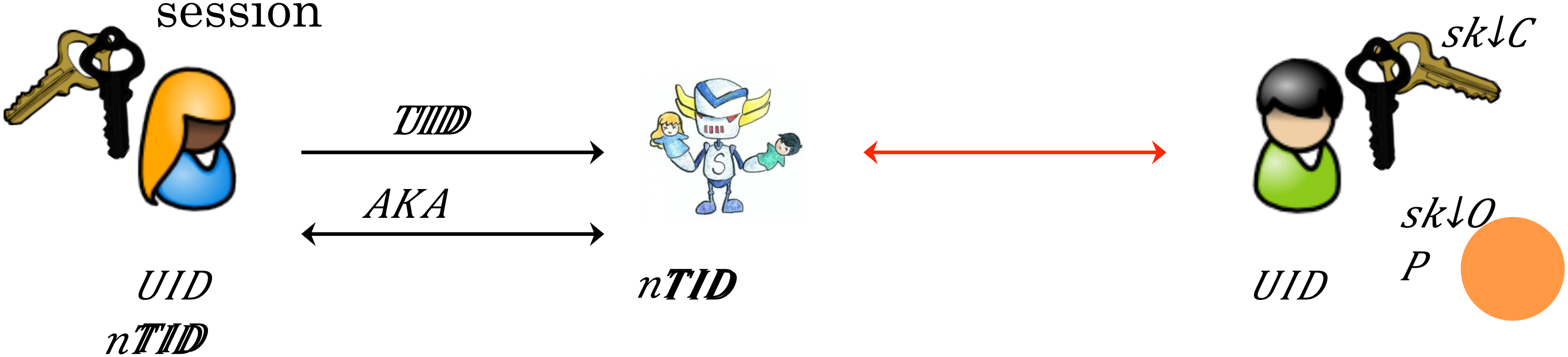
    No PKI for Terminals (too many of them)



1001 IDENTIFIERS

- Client associated with secret keys: sk_C, sk_{OP}, st_C
  - All clients of the same operator share same sk_{OP}

- Other identifiers:
  - Operator associates C with unique UID (permanent)
  - Each terminal T_i associates C with 4B TID (temporary), unique per terminal, updated per session

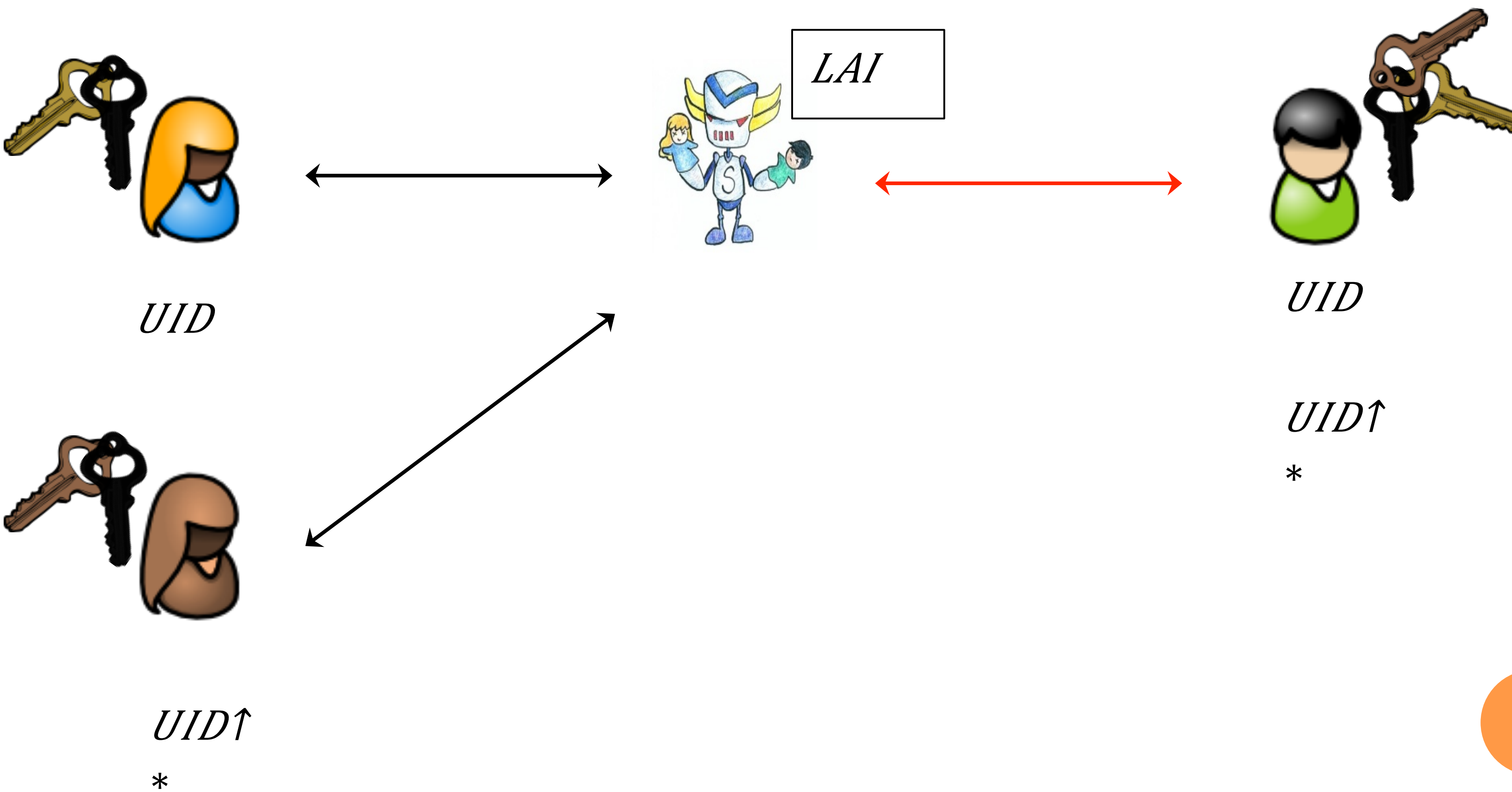


1001 Identifiers (contd.)

- Each terminal has non-colliding list of *TID*s
  - Inter-terminal collisions possible
  - No “centralized” DB of all *TID*s
- Each terminal is associa-ted with unique *LAI*
  - Like ZIP code
  - (*TID*, *LAI*) unique

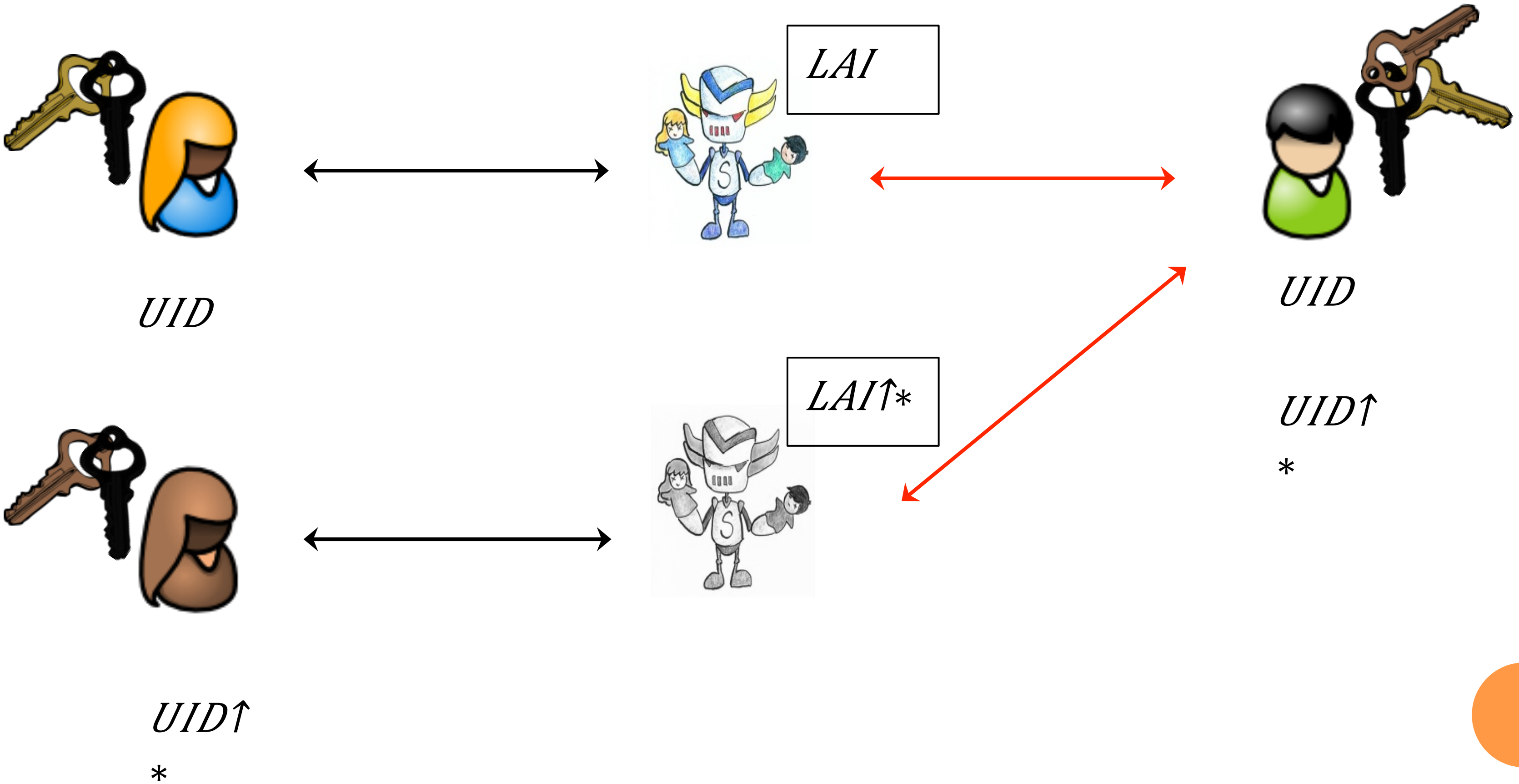
1001 Identifiers (summary)

- Multiple clients of same Operator



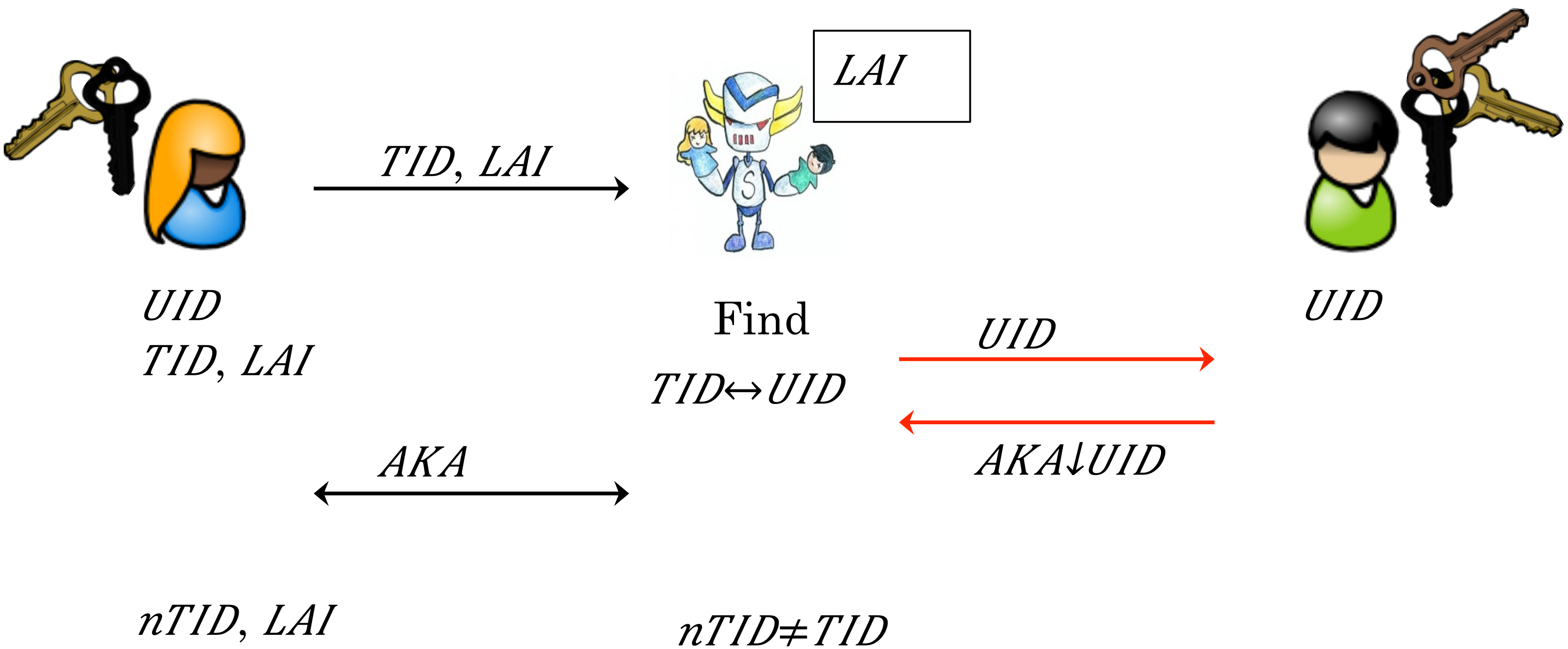
1001 IDENTIFIERS (SUMMARY)

- Multiple clients of same Operator



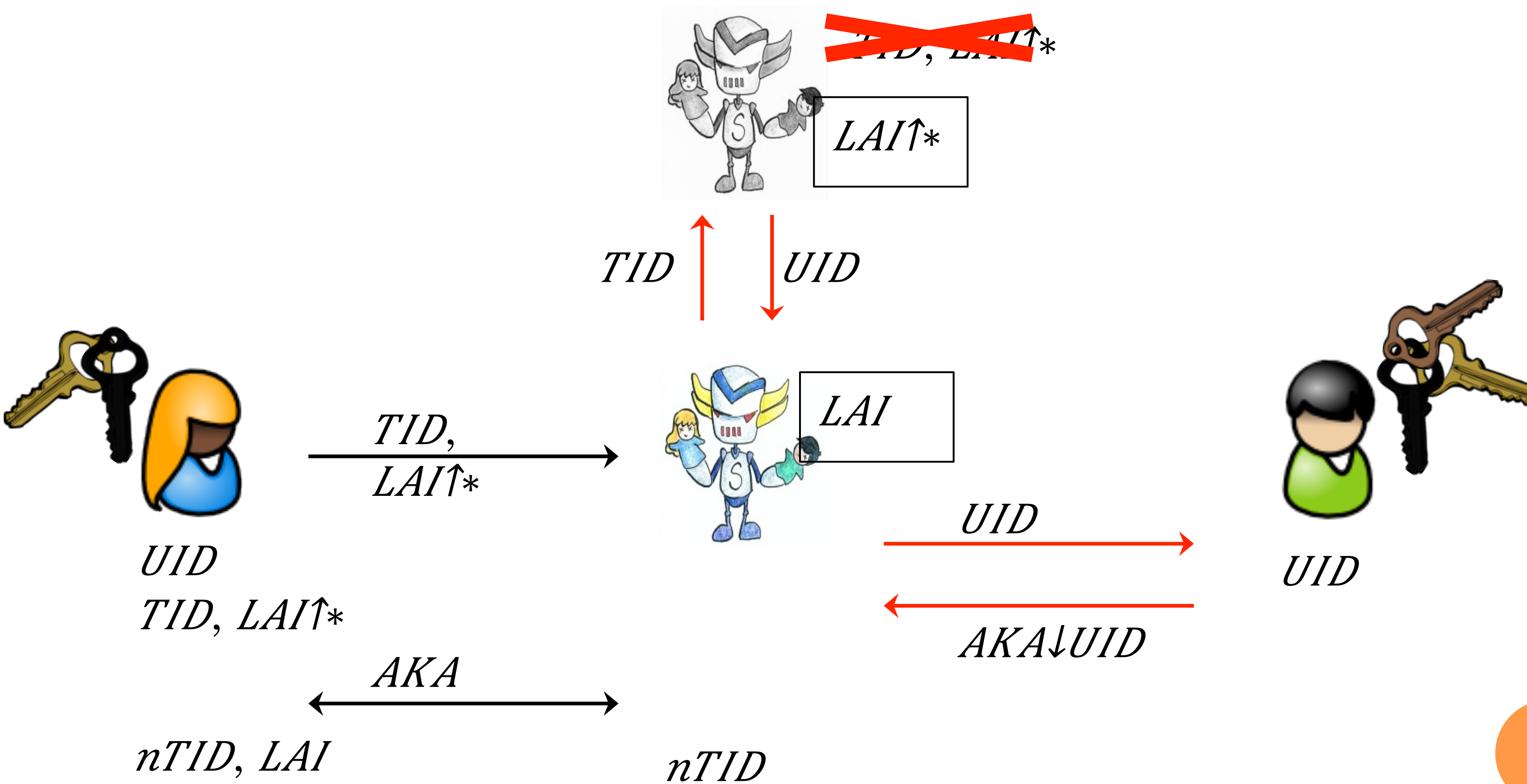
1001 Identifiers (summary)

- *TID* and *UID* in protocol run, same *LAI*



1001 IDENTIFIERS (SUMMARY)

- *TID* and *UID* in protocol run, different *LAI*



Possible (not likely)
nTID=TID

Secret Keys, Secret State

- Client associated with secret keys: sk_C, sk_{OP}, st_C
  - All clients of the same operator share same sk_{OP}
- State st_C is a sequence number
  - Terminal also has a state st_{OP}^C w.r.t. that client
  - Used as “shared” randomness for authentication
  - Initially randomly chosen for each client
  - Then updated by update function (3 possibilities)
  - Unlike sk_{OP}, sk_C, Terminals may know st_C

Part III. 2
Underlying Cryptography

Cryptographic Functions

- The seven dwarfs:

  F_1 : used by terminal, for terminal authentication
  input $(sk_C, sk_{OP}, R, Sqn_{OP}^C, AMF)$

  F_1^* : used by client in special procedure
  input $(sk_C, sk_{OP}, R, Sqn_{OP}^C, AMF)$

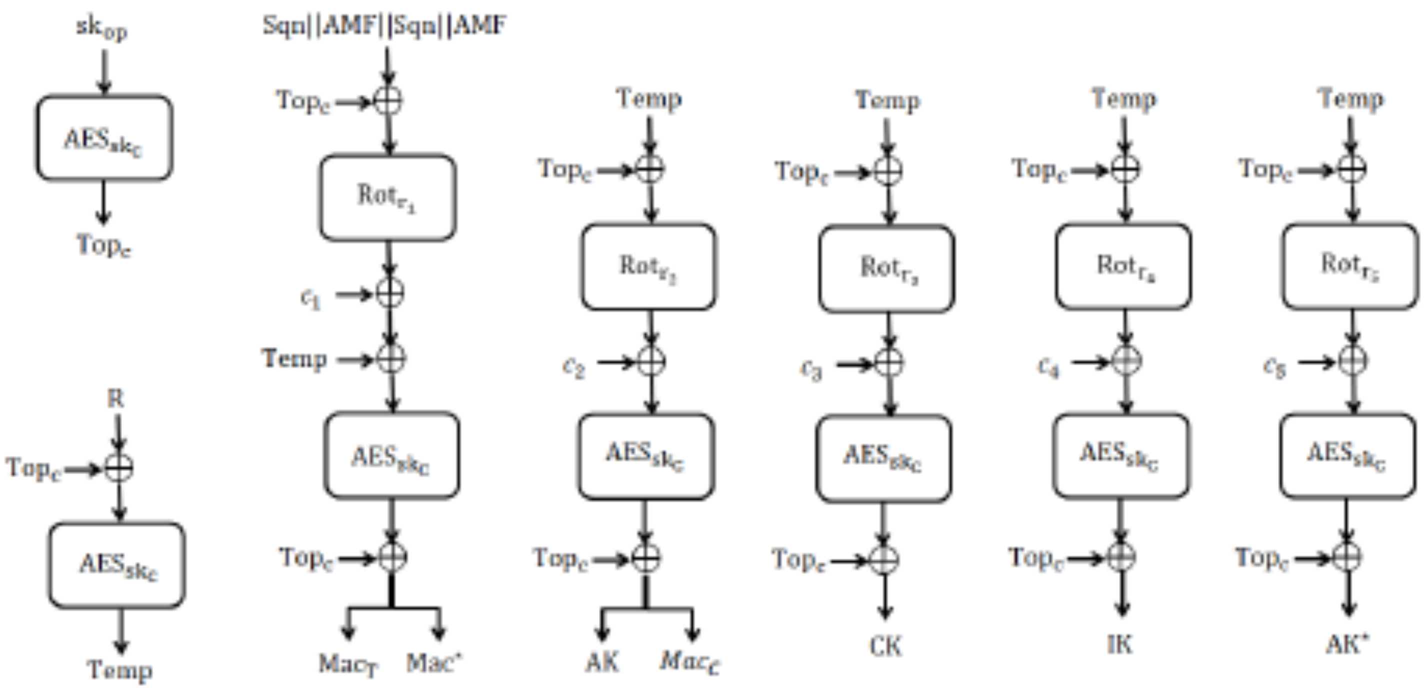
  F_2 : used by client, for client authentication
  input (sk_C, sk_{OP}, R)

  F_3, F_4 : used by both for session-key generation
  input (sk_C, sk_{OP}, R)

  F_5 : used by terminal for "blinding" key
  input (sk_C, sk_{OP}, R)

  F_5^* : used by client for "blinding" key, special procedure
  input (sk_C, sk_{OP}, R)

MILENAGE



F↓1
F↓1↑
*

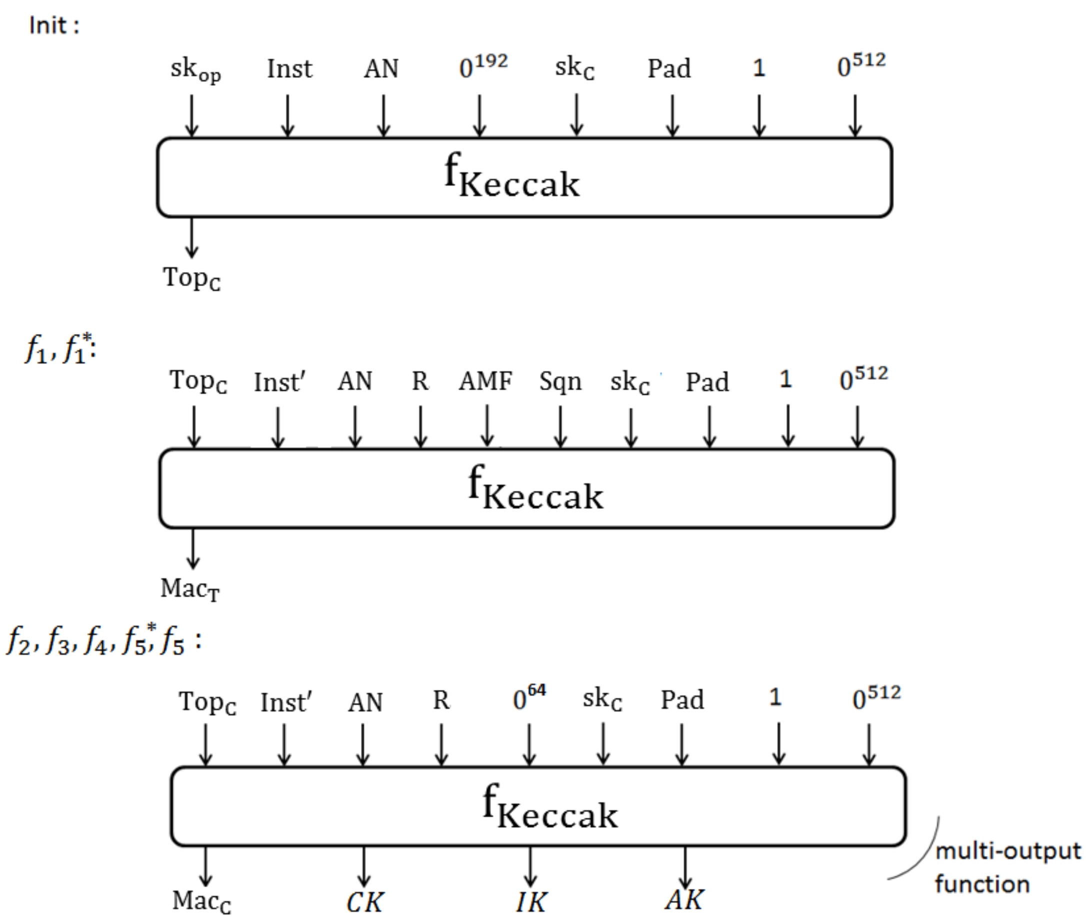
F↓5
F↓2

F↓3

F↓4

F↓5↑
*

TUAK



F↓2 *F*↓3 *F*↓4 *F*↓5 , *F*↓5↑

What we Proved for TUAK

- Single function *G* generalizing the seven functions

TUAK: $G\downarrow TUAK$ is PRF assuming that the internal permutation of Keccak is PRF

Stronger than "each function is PRF"!

- Intuition of $G\downarrow TUAK$: use handy truncation of output

WHAT WE PROVED FOR MILENAGE

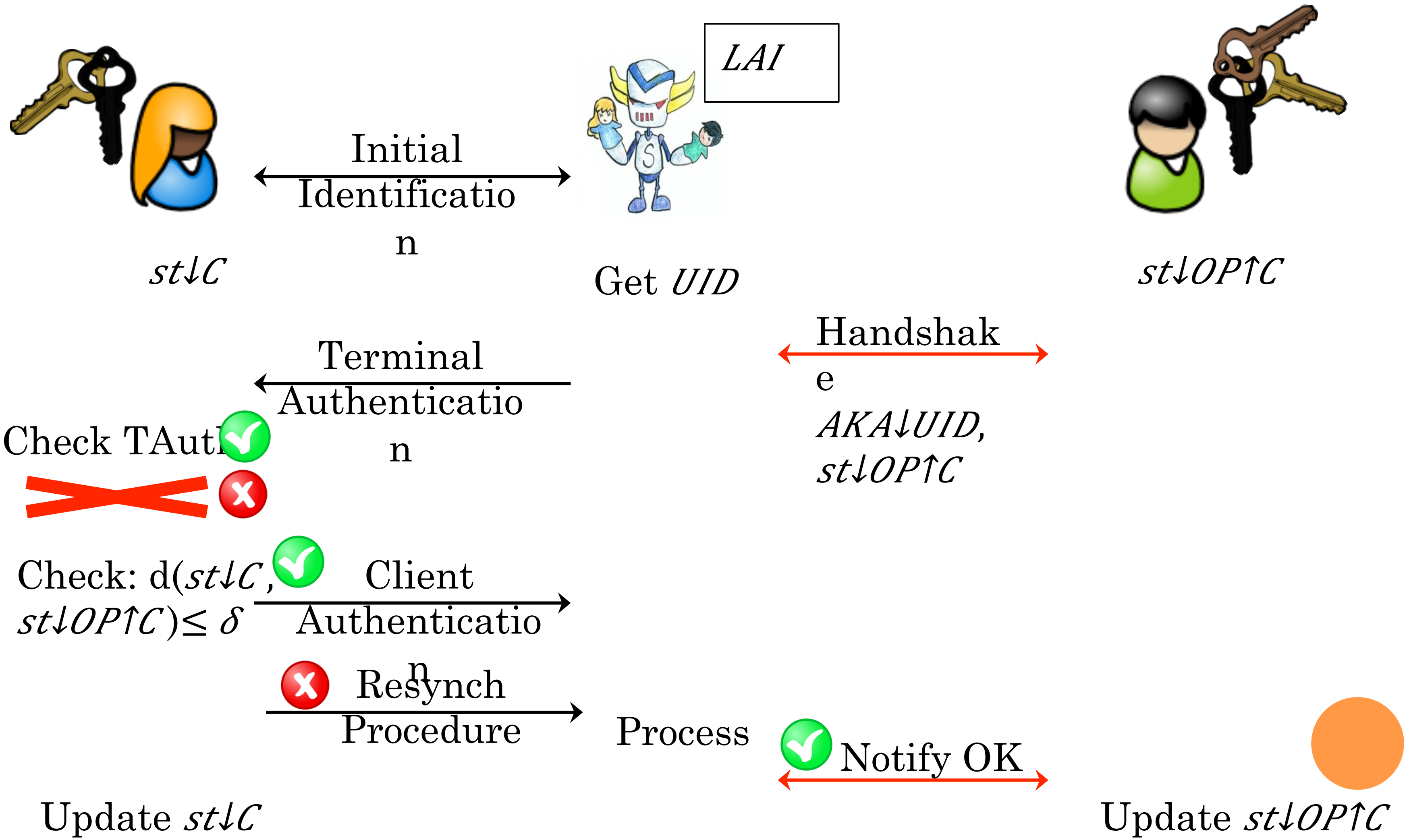
- Single function G generalizing the seven functions
  - Becomes 2 functions never used together in same call

MILENAGE: G_{MIL}, G'_{MIL} are PRF assuming that the AES permutation is PRF

- Intuition of $G_{MILENAGE}$: use handy XOR-ing in all the right places

Part III. 3
The Protocol

AKA STRUCTURE (BASIC)



AKA Structure (Real)

Initial Identification

LAI

Get *UID*

st_{OP}^{C}

Handshakes

AKA_{UID}, st_{OP}^{C}

AKA_{UID}, $Update[st_{OP}^{C}]$

…

AKA_{UID}, $Update^{N}[st_{OP}^{C}]$

Update st_{C}

AKA_{UID}, st_{OP}^{C}

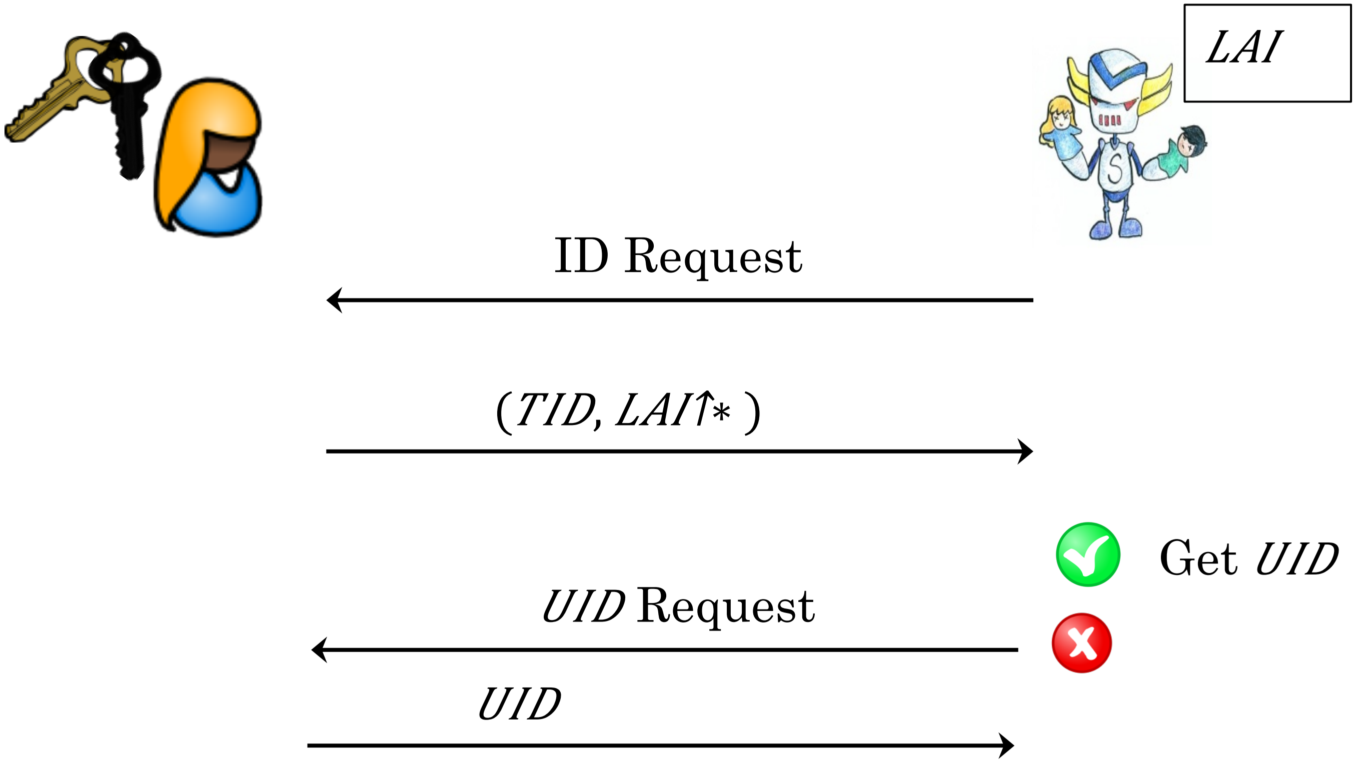
AKA_{UID}, $Update[st_{OP}^{C}]$

Update st_{C}

OK $Update[st_{OP}^{C}]$

Update st_{OP}^{C}

INITIAL IDENTIFICATION



Fundamental privacy flaw: *UID* easily obtainable!

Security: even if *UID* replaced, still OK (authentication automatically fails)

HANDSHAKE PREPARATION (1 BATCH)

LAI

UID →

Set $Sqn = Update[st_{OP}^{C}]$
Generate R at random.

Compute:

$MAC_{OP} = F_1(sk_C, sk_{OP}, R, Sqn, AMF)$

$MAC_C = F_2(sk_C, sk_{OP}, R)$

Reveals Sqn → $AK = F_5(sk_C, sk_{OP}, R)$

$Autn = (Sqn \text{ XOR } AK) \,||\, AMF \,||\, MAC_{OP}$

$CK = F_3(sk_C, sk_{OP}, R)$

← R and everything

$IK = F_4(sk_C, sk_{OP}, R)$

Terminal/Client AKE

LAI

$R \,||\, (Sqn \text{ XOR } AK) \,||\, AMF \,||\, MAC_{OP}$

←

Compute: $AK = F_5(sk_C, sk_{OP}, R)$
Retrieve Sqn and check MAC_{OP}

If $Sqn \in \{st_C, \ldots, st_C + \delta\}$

Compute:
$Rsp = F_2(sk_C, sk_{OP}, R)$
$CK = F_3(sk_C, sk_{OP}, R)$
$IK = F_4(sk_C, sk_{OP}, R)$

Rsp →

Check $Rsp = MAC_C$

Else Resynch!

Resynch Procedure

LAI

If MAC_{OP} verifies, but *Sqn* out of range
Compute:

$AK^* = F_5^*(sk_C, sk_{OP}, R)$
$MAC_C^* = F_1^*(sk_C, sk_{OP}, st_C, AMF, R)$

$(st_C \text{ XOR } AK^*) \,||\, MAC_C^*$

Compute: AK^*, get st_C
Check: out of range
Check: MAC_C^*
Set $st_{OP}^C = st_C$

Start from there.

Part IV
Security of AKA

Cleaner Abstraction

C		T
$(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{st_C} = \mathsf{Sqn_C})$		$(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{st_{T,C}} = \mathsf{Sqn_{T,C}})$
	← UID Request	
	UID →	
		Generate R. Denote: $\mathsf{Sqn} \leftarrow \mathsf{Up}(\mathsf{Sqn_{T,C}})$ Compute: $\mathsf{Mac_T} \leftarrow \mathcal{F}_1(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R}, \mathsf{Sqn}, \mathsf{AMF})$, $\mathsf{Mac_C} \leftarrow \mathcal{F}_2(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R})$, $\mathsf{AK} \leftarrow \mathcal{F}_5(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R})$, $\mathsf{Autn} \leftarrow (\mathsf{Sqn} \oplus \mathsf{AK}) \Vert \mathsf{AMF} \Vert \mathsf{Mac_T}$.
	← R‖Autn	
Compute AK using R. Recover Sqn (from AK). Check $\mathsf{Mac_T}$ value. If $\mathsf{Sqn} \in (\mathsf{Sqn_C}, \mathsf{Sqn_C} + \Delta)$: Compute: $\mathsf{CK} \leftarrow \mathcal{F}_3(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R})$, $\mathsf{IK} \leftarrow \mathcal{F}_4(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R})$, Set $\mathsf{Res} := \mathcal{F}_2(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R})$. Update $\mathsf{st_C} := \mathsf{Sqn}$. Else *re-synchronization*		
	Res →	
		Iff. $\mathsf{Res} == \mathsf{Mac_C}$, set: $\mathsf{CK} \leftarrow \mathcal{F}_3(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R})$, $\mathsf{IK} \leftarrow \mathcal{F}_4(\mathsf{sk_C}, \mathsf{sk_{op}}, \mathsf{R})$, Update: $\mathsf{Sqn_{T,C}} \leftarrow \mathsf{Sqn}$.

Security Properties

Key Secrecy: Attained under assumption of pseudorandomness of G

Advantage is linear in number of clients!

Client Impersonation: Attained under assumption of pseudorandomness of G

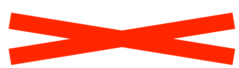
Advantage is linear in $N_C / 2^{|MAC_C|}$ and $N_C / 2^{|sk_C|}$

Offline Relays

LAI

Initial Identification

Get UID



$R \,||\, (Sqn \text{ XOR } AK) \,||\, AMF \,||\, MAC\downarrow OP$

Initial Identification

$R \,||\, (Sqn \text{ XOR } AK) \,||\, AMF \,||\, MAC\downarrow OP$

Terminal-Impersonation Resistance

- Attained as long as there are no offline relays
  - Thus weaker than Client Impersonation guarantee

Terminal Impersonation: Attained under assump-tion of pseudorandomness of G

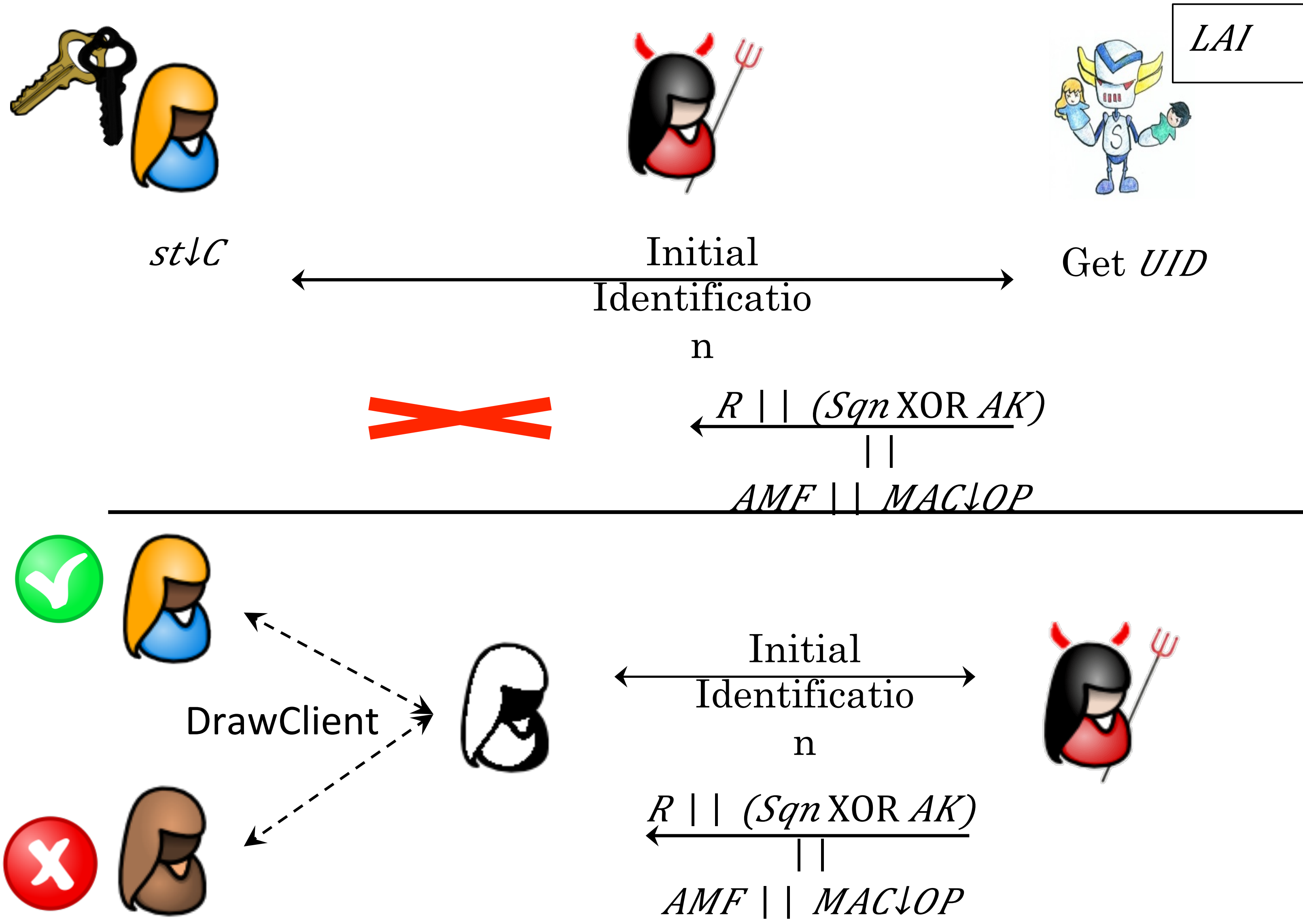
Advantage is linear in $N_C / 2^{|MAC_C|}$ and $N_C / 2^{|sk_C|}$

Part V
Lack of Privacy & Impossibilities

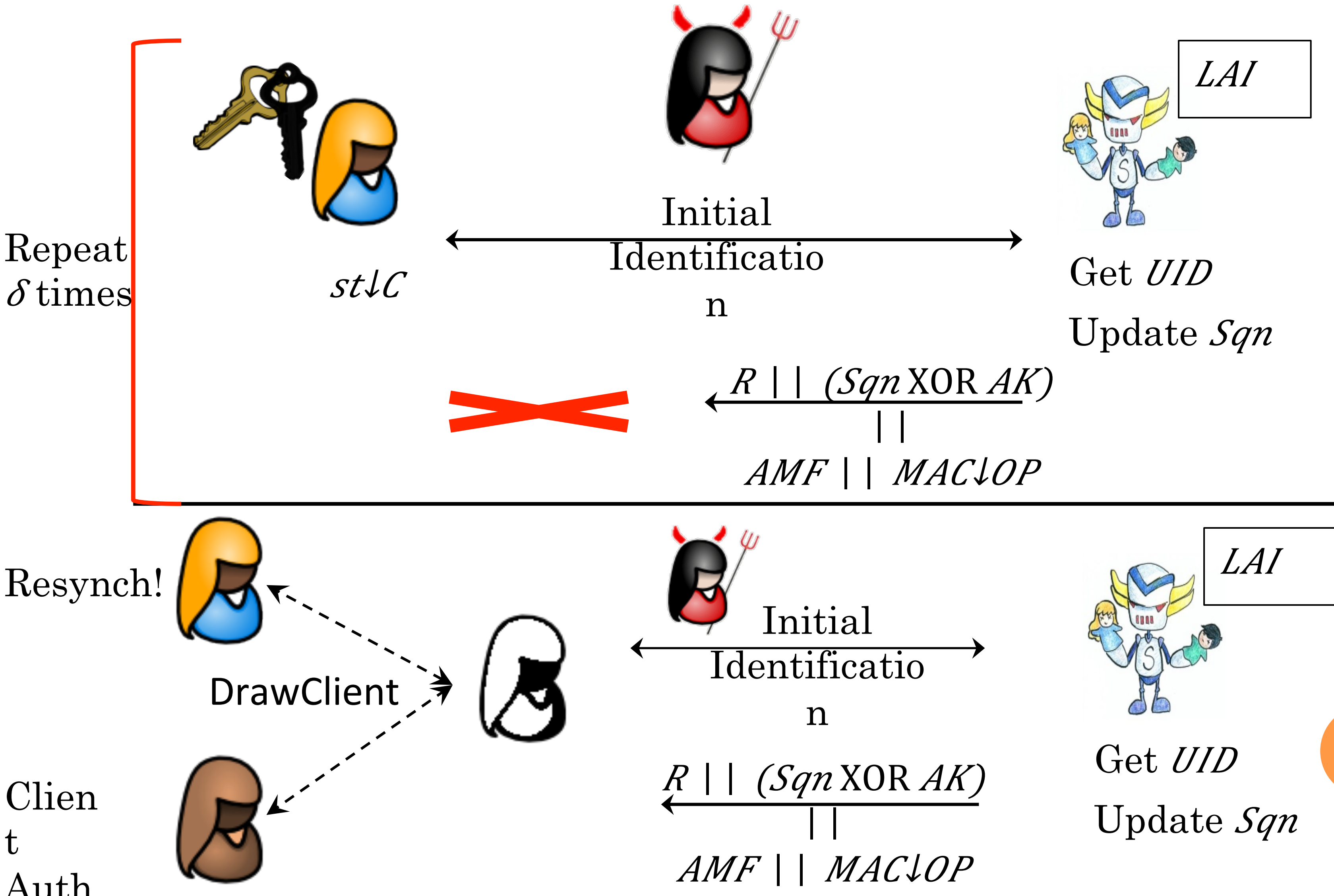
Truth or Dare

- 3 GPP claim AKA is:
  - ID-Hiding – nobody can identify client
  - Location-hiding – nobody can trace client location
  - Untraceable – nobody can link client sessions
- Their arguments:
  - Nobody knows *UID* and normally it is not used
  - Sequence number and keys are hidden in transcripts
- We PROVE AKA is:
  - NOT ID-Hiding – very easy to recover *UID*
  - Location-hiding – not really…
  - NOT Untraceable – see some attacks next slide

Distinguishing by Terminal Impersonation



Distinguishing By Resynchronization



The Big Impossibility

- Goal: make this protocol forward-private
  - Trivial solution: just build a new protocol
  - How do we do it without changing it too much?
- The past kills your future
  - In the AKA protocol the client is always one step behind the terminal/operator in state.
  - Client corruption at time t is enough to identify client in $(t-1)$st transcript
  - Generalizable attack: problem is that 3GPP do not want the client to choose anything

But we can fix the protocol to get weak privacy

Part VI
Conclusions

AKE Protocols

- Authenticated Key Exchange
  - Goal: construct a secure channel between two parties
  - 2 steps:
    - Handshake: derive keys for authenticated encryption
    - Use the keys to encrypt and sign your communication
- Examples: TLS/SSL, AKA
- Authentication:
  - Unilateral : only one party authenticates the keys
  - Mutual: both parties authenticate the keys

Theory vs. Practice & AKA

- AKA: symmetric-key AKE protocol with mutual authentication
- 1001 identifiers, all more or less secret, some temporary and some not
- 2 algorithm suites:
  - Milenage: based on AES
  - TUAK: based on Keccak
- Security:
  - Key Secrecy, Client Impersonation & some terminal impersonation security
- Privacy: many attacks, impossible to really fix for stronger privacy notions