

# Malicious Participants in GKE

## *Key Control and Contributiveness in the Shadow of Trust*



Emmanuel Bresson

DCSSI Crypto Lab

Paris, France

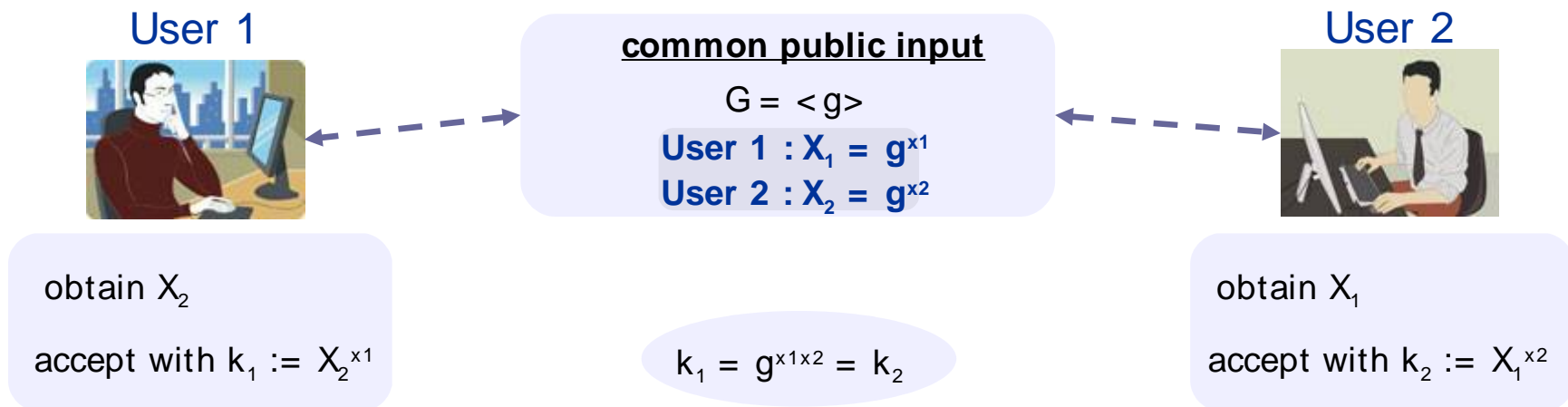
Mark Manulis

Horst Görtz Institute for IT Security

Bochum, Germany

# “Genuine,, Diffie- Hellman Key Exchange<sup>[DH76]</sup>

- 2- party key exchange protocol
- foundational for many group key exchange protocols<sup>[ITW82,SSDW88,BD94,...]</sup>
- computations are performed in the *finite cyclic* group  $G$ 
  - $g$  is the generator of  $G$
  - *Discrete Logarithm Problem* (given  $g^x$  find  $x$ ) is intractable in  $G$



# „Referenced“ Diffie- Hellman Key Exchange

- users choose own secret exponents during the protocol execution

User 1



choose random  $x_1$

$$X_1 := g^{x_1}$$

$X_1$  →

←  $X_2$

accept with  $k_1 := (X_2)^{x_1}$

User 2



choose random  $x_2$

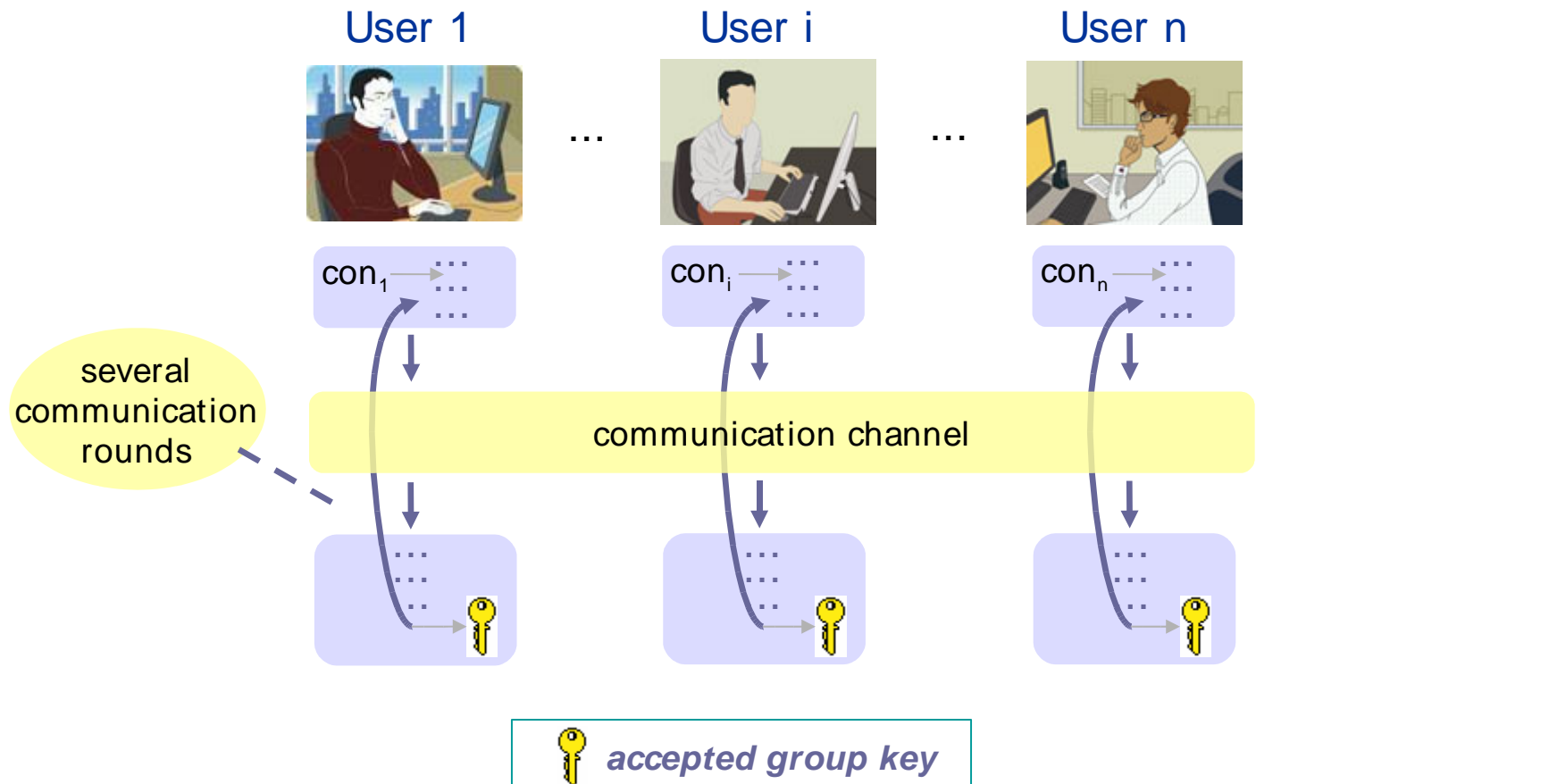
$$X_2 := g^{x_2}$$

←  $X_2$

$X_1$  →

accept with  $k_2 := (X_1)^{x_2}$

# Group Key Exchange (GKE)



# Security Threats in GKE



malicious participants

[KS05,CHB05]

(passive/ active)  
adversarial control  
of communication

[BR93,BCK98,CK01,KY03]

User 1



User i



User n



honest participants  
with revealed  
internal states  
(strong corruptions)

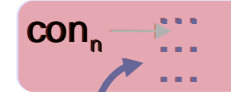
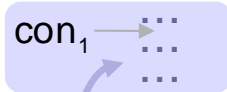
[S99,BPR00,CK01,BCP02]

passive A

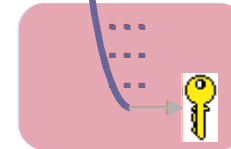
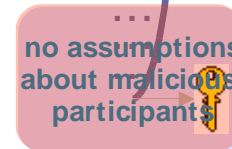
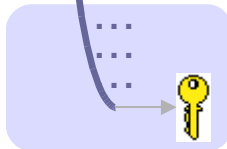
eavesdrop  
drop  
delay  
change order

active A

as passive +  
replay  
modify  
inject



asynchronous unreliable communication channel



# Security Remarks for „Genuine“ DH- KE

User 1



obtain  $X_2$

accept with  $k_1 := X_2^{x_1}$

$$k_1 = g^{x_1 x_2} = k_2$$

User 2



obtain  $X_1$

accept with  $k_2 := X_1^{x_2}$

- every new session results in the same key
  - ⊖ no key secrecy if other session keys are exposed (known-key security)<sup>[B94]</sup>
- long-term keys  $(x_1, x_2)$  used directly to compute the key
  - ⊖ no key secrecy if  $(x_1, x_2)$  are exposed later (weak forward secrecy)<sup>[G89]</sup>
- long-term keys are linked to the users' identities
  - ⊕ adversary cannot act on behalf of the users (impersonation resilience)<sup>[BD94]</sup>

# Security Remarks for „Reference“ DH- KE

User 1



choose random  $x_1$

$$X_1 := g^{x_1}$$

$$\xrightarrow{X_1}$$

$$\xleftarrow{X_2}$$

accept with  $k_1 := (X_2)^{x_1}$

User 2



choose random  $x_2$

$$X_2 := g^{x_2}$$

$$\xleftarrow{X_2}$$

$$\xrightarrow{X_1}$$

accept with  $k_2 := (X_1)^{x_2}$

- session keys are independent
  - ⊕ known-key security is provided
- no long-term keys are used
  - ⊕ weak forward secrecy is provided, *but*
  - ⊖ impersonation attacks become possible
- ephemeral secrets  $(x_1, x_2)$  are used to compute the key
  - ⊖ no key secrecy if  $(x_1, x_2)$  are exposed later (strong forward secrecy) [BPR00, CK01]

# Achieving Strong Forward Secrecy

- Idea: *erase ephemeral secrets* prior to acceptance, e.g., *secure erasure*<sup>[CFJ99]</sup>



- ephemeral secrets used to compute the key are erased  
+ strong forward secrecy

- no long-term keys are used  
+ weak forward secrecy is provided,  
- *but* impersonation attacks still possible

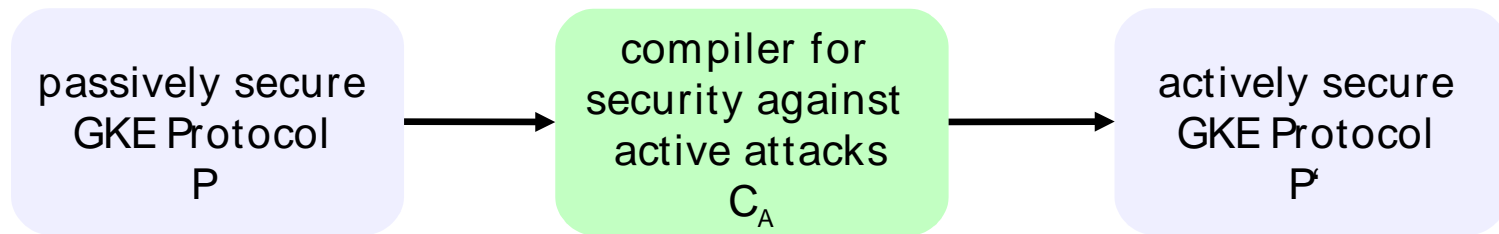
# Generic Solution against Active Attacks

## Passive Attacks

drop, delay, change order<sub>[CK01]</sub>

## Active Attacks

replay, modify, inject



## Building Blocks

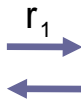
- *digital signature scheme* (Gen, Sign, Verify)
  - every **User**  $i$  holds a long-term key pair  $(sk_i, pk_i) \leftarrow \text{Gen}$
  - every  $pk_i$  is publicly known and linked to **User**  $i$
  - *provides existential unforgeability and non-repudiation*

# $C_A$ : Modified Katz - Yung Compiler<sup>[KY03]</sup>

User 1



choose random  $r_1$   
 $sid_1 = r_1|r_2$



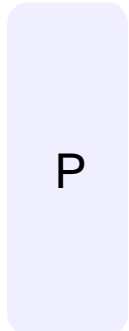
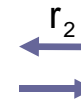
$pid_1 = \text{User 1} | \text{User 2} = pid_2$

computation of the unique session id

User 2



choose random  $r_2$   
 $sid_2 = r_1|r_2$

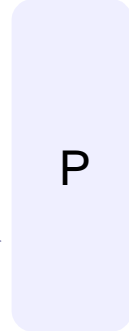


$m \rightarrow \sigma_1 = \text{Sign}(sk_1, m|sid_1|pid_1)$

$m|\sigma_1$

$\text{Verify}(pk_1, m|sid_2|pid_2, \sigma_1)$

$m$



$k_1$

accept with  $k_1$

$k_2$

accept with  $k_2$

# $C_A$ -compiled „Referenced“ DH- KE

User 1



$sid_1 = r_1 | r_2$

choose random  $x_1$

$X_1 := g^{x_1}$

$k_1 := (X_2)^{x_1}$

erase  $x_1$

accept with  $k_1$

computation of the unique session id

$X_1 | \sigma_1$

$X_2 | \sigma_2$

asynchronous unreliable  
communication channel

$X_2 | \sigma_2$

$X_1 | \sigma_1$

User 2



$sid_2 = r_1 | r_2$

choose random  $x_2$

$X_2 := g^{x_2}$

$k_2 := (X_1)^{x_2}$

erase  $x_2$

accept with  $k_2$

- Recall: malicious participants may deviate from the protocol specification and internal states of honest participants may be revealed
- malicious users can *exclude contributions* of honest users upon computing  $k$  (key control<sup>[M98]</sup>, contributiveness<sup>[AST98]</sup>, key replication<sup>[K05]</sup>)

# Attack against Contributiveness

- malicious User 1 wishes that User 2 accepts  $k^* = g^{x^*}$  for some chosen  $x^*$



sid<sub>1</sub> = r<sub>1</sub>|r<sub>2</sub>

sid<sub>2</sub> = r<sub>1</sub>|r<sub>2</sub>

wait

$x_1 := x^* / x_2$

$X_1 := g^{x_1}$

$k_1 := (X_2)^{x_1}$

erase  $x_1$

choose random  $x_2$

$X_2 := g^{x_2}$

$k_2 := (X_1)^{x_2}$

erase  $x_2$

asynchronous unreliable  
communication channel

$X_2 | \sigma_2$

$X_1 | \sigma_1$

$X_2 | \sigma_2$

$X_1 | \sigma_1$

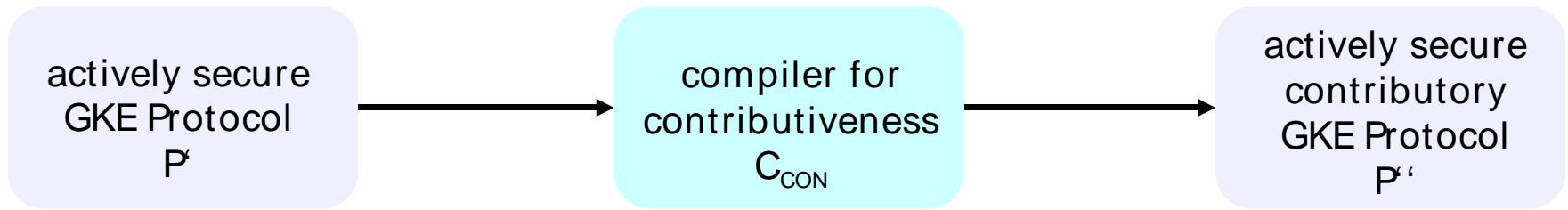
reveal

accept with  $k_1 = k^*$

accept with  $k_2 = k^*$

- malicious User 1 succeeds for **any** choice of  $x_2$  in **any** protocol session

# Generic Solution for Contributiveness



## Building Blocks

- *collision-resistant pseudo-random function*  $f(s, v)$ 
  - $s$  – uniformly chosen secret seed,  $v$  – (public) input value
  - *collision-resistance*
    - for all  $s \neq s'$  holds  $f(s, v) \neq f(s', v)$
  - *pseudo-randomness*
    - $v \rightarrow f(s, v)$  indistinguishable from a random function
- *one-way permutation*  $\pi$ 
  - given  $\pi(x)$  it is infeasible to find  $x$

# Security Compiler $C_{\text{CON}}$

User 1



$P'$

$k_1 \downarrow$

choose random  $r_1$

$\text{sid}_1 = r_1 | r_2$

$\rho_1 := f(k_1 \oplus \pi(r_1), v)$

$\rho_2 := f(\rho_1 \oplus \pi(r_2), v)$

$K_1 := \rho_2$

erase  $k_1, \rho_1, \rho_2$

accept with  $K_1$

common public input

$v$

actively secure GKE protocol

computation of the unique session id

iterative embedding of  $r_i$   
no further communication is needed

User 2



$P'$

$k_2 \downarrow$

choose random  $r_2$

$\text{sid}_2 = r_1 | r_2$

$\rho_1 := f(k_2 \oplus \pi(r_1), v)$

$\rho_2 := f(\rho_1 \oplus \pi(r_2), v)$

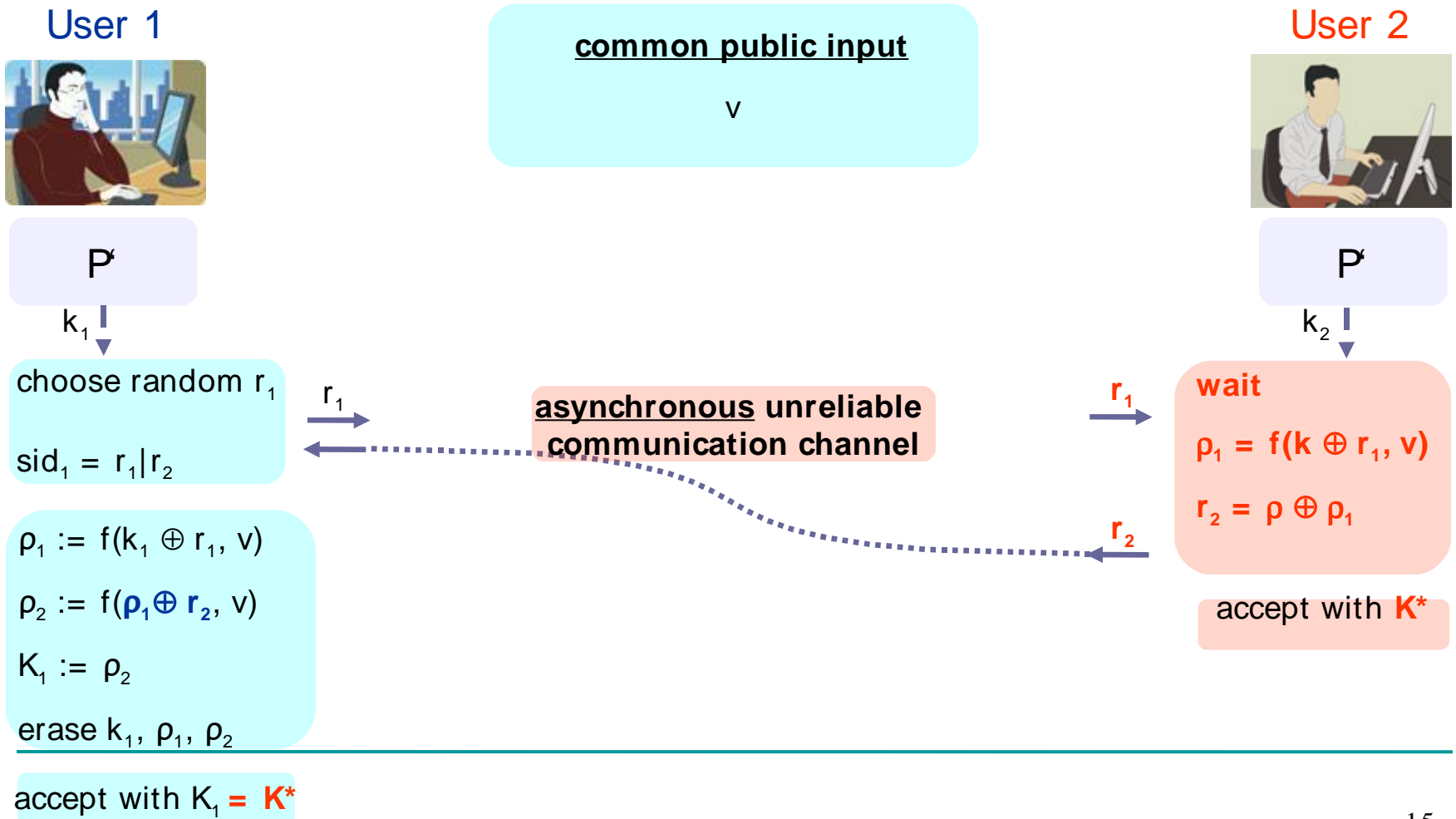
$K_2 := \rho_2$

erase  $k_1, \rho_1, \rho_2$

accept with  $K_2$

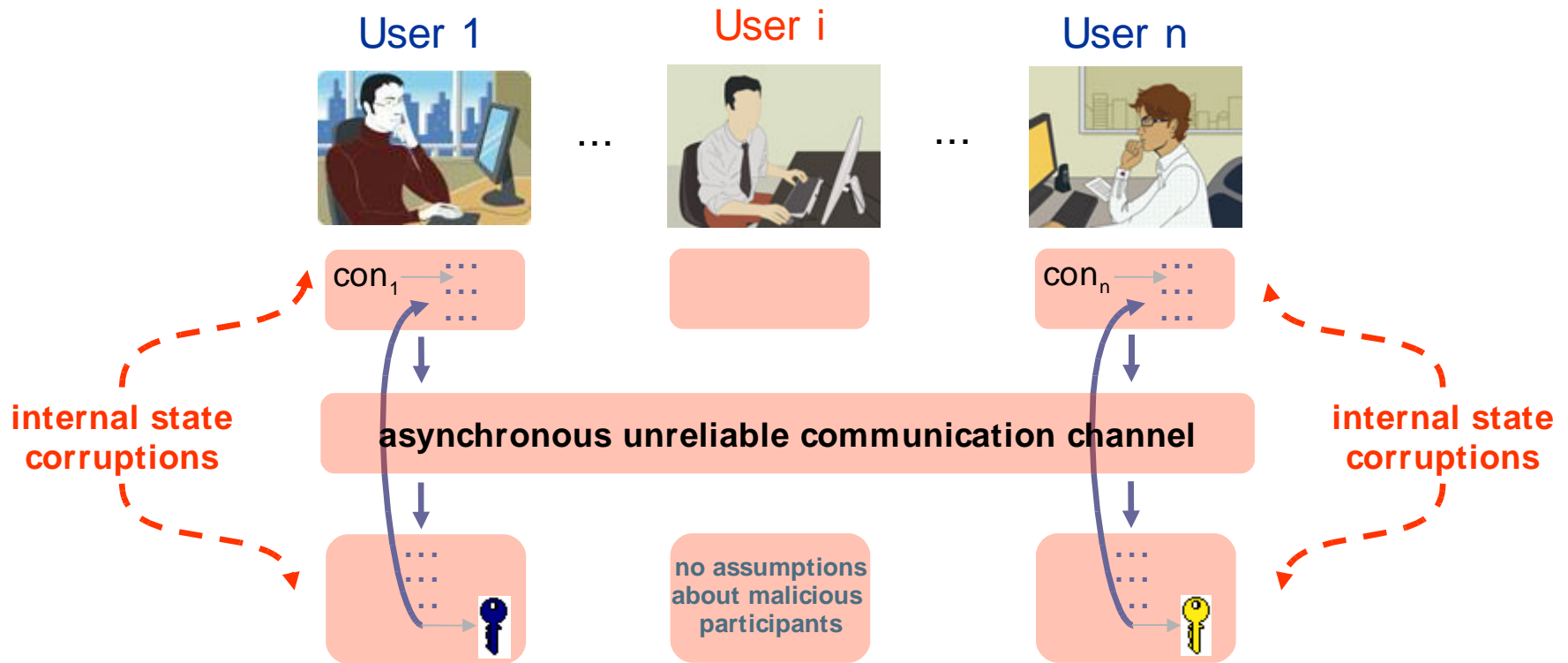
# Why do we need $\pi$ ?

- assume that  $\rho_i = f(\rho_{i-1} \oplus r_i, v)$  with  $\rho_0 = k$
- malicious User 2** chooses  $K^* = f(\rho, v)$  for *some chosen*  $\rho$



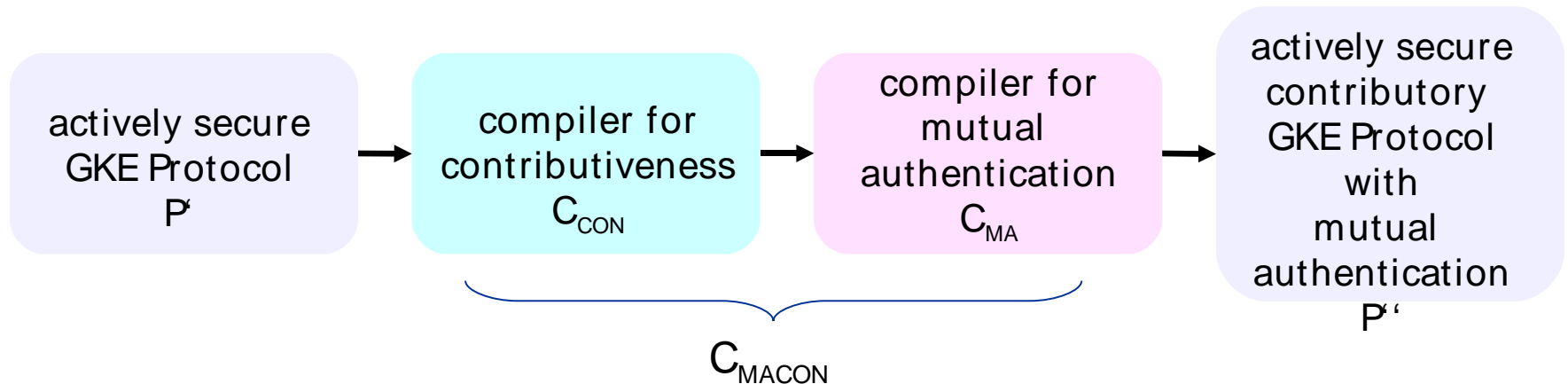
# Mutual Authentication Security

- key confirmation<sup>[MOV96]</sup>: all honest users who accept<sup>[S99]</sup> hold *identical* keys



- mutual authentication<sup>[BR93]</sup>: all honest users who accept<sup>[S99]</sup> are *aware* of each other participation in the protocol

# Generic Solution for Mutual Auth.

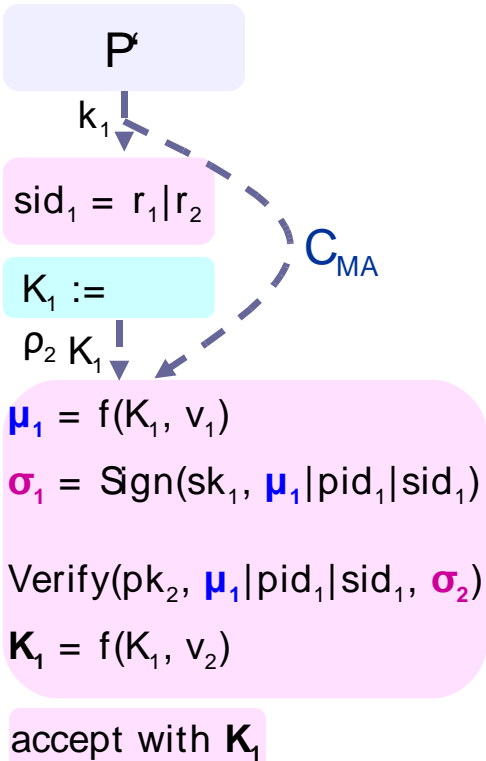


## Building Blocks

- *collision-resistant pseudo-random function  $f(s, v)$* 
  - similar to  $C_{CON}$
- *digital signature scheme (Gen, Sign, Verify)*
  - similar to  $C_A$

# Security Compiler $C_{\text{MACON}}$

User 1



common public input

$v$

User 1 :  $pk_1$

User 2 :  $pk_2$

$v_1, v_2$

$\text{pid}_1 = \text{User 1} | \text{User 2} = \text{pid}_2$

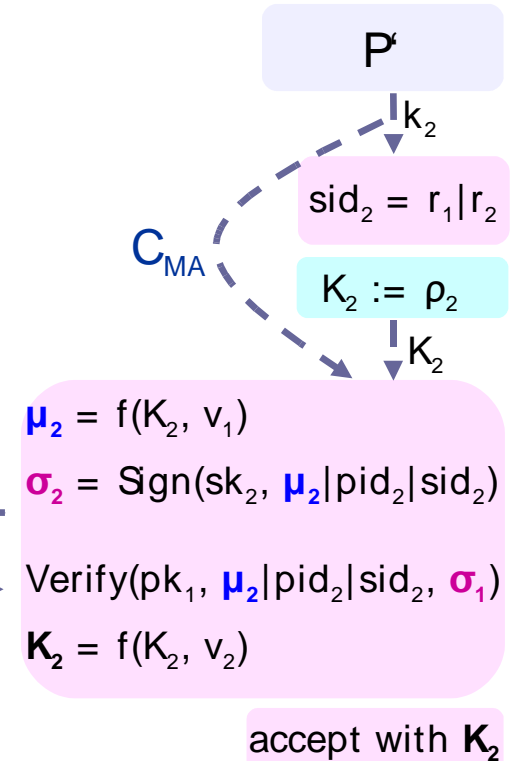
actively secure GKE protocol

computation of the unique session id

iterative embedding of  $r_i$

key confirmation tokens

User 2



# Summary

- security- enhancing compiler  $C_{\text{MACON}}$ 
  - contributiveness, key control, key replication
  - key confirmation, mutual authentication, unknown- key share
  - considers malicious participants and strong corruptions
- extended version appears at ATC 2007
- in the full paper
  - formalized security definitions
  - (game- based) security proofs

Thank You!