

On Security Models and Compilers for Group Key Exchange Protocols

Emmanuel Bresson

DCSSI Crypto Lab
Paris, France

Mark Manulis

UCL Crypto Group
Louvain-la-Neuve, Belgium

Jörg Schwenk

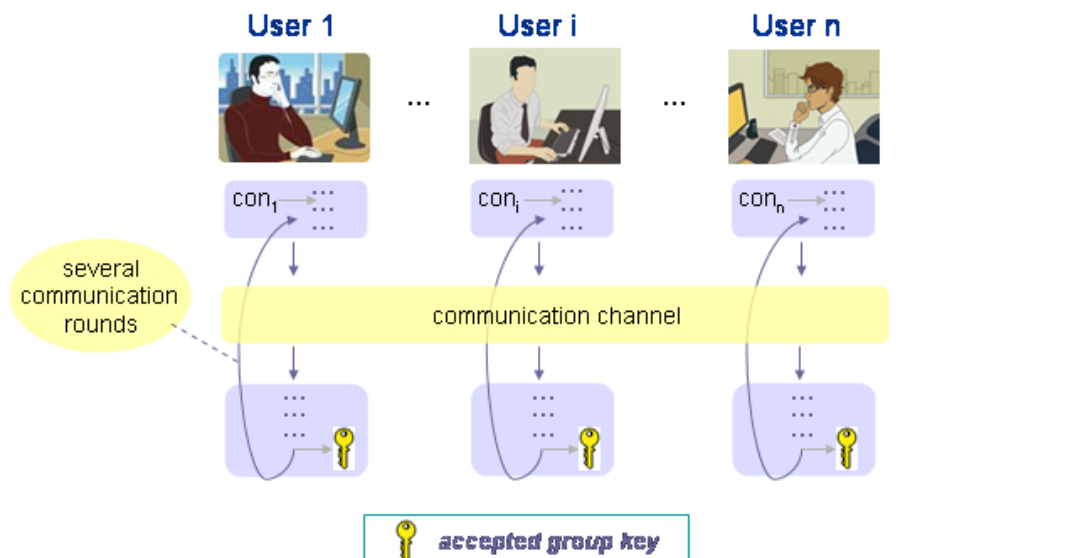
Horst Görtz Institute for IT Security
Bochum, Germany

Oct 29-31, Nara (Japan)

International Workshop on Security 2007

1

Group Key Exchange (GKE)



con_i secret contribution of User i

Oct 29-31, Nara (Japan)

International Workshop on Security 2007

2

Core Security Requirements for GKE

- **Authenticated Key Exchange (AKE)**
 - indistinguishability of the key from a random
 - known-key security
 - forward secrecy (corruptions in the future should keep secrecy of session keys in the past)
- **Mutual Authentication (MA)**
 - assurance of participation
 - key confirmation

Security Models for GKE

- needed to provide *provable security*
- **model protocol execution**
 - participants
 - sessions (concurrent/sequential executions)
- **model security requirements**
 - adversarial capabilities
 - adversarial goals (games)

Participants

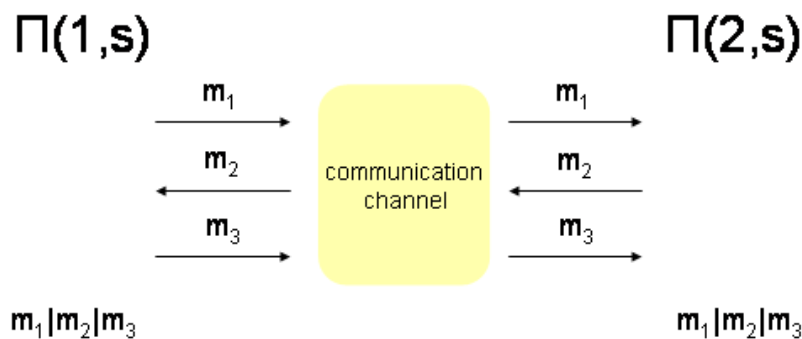
- **U** is a set of n *users* in the universe
- every $U_i \in U$ has a long-lived key LL_i
- not enough for concurrent sessions

Sessions

- unlimited number of *instances* $\Pi(i,s)$ for U_i
- every instance becomes associated with
 - *session group key* $k(i,s)$
 - *session id* $sid(i,s)$
 - *partner id* $pid(i,s)$

how to identify instances of the same session?

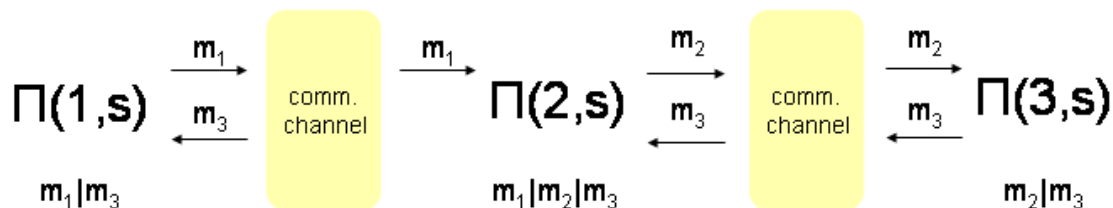
Matching Conversations^[BR93]



- **existence of matching conversations ensures**
 - participation of the instances in the execution, thus instances are *partnered*
 - acceptance of the instances with identical keys
- **technical core of MA-security in [BR93]**

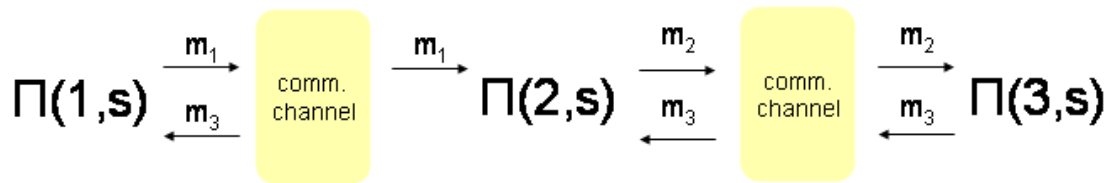
Matching Conversations in GKE?

- **MCs are not sufficient for all GKE (i.e., only for broadcast GKEs as in [KY03])**



- **attempt to extend MCs to groups in [BCPQ01]**
- **definition of the *partnering condition* through *partially* matching conversations**

Session IDs in BCPQ



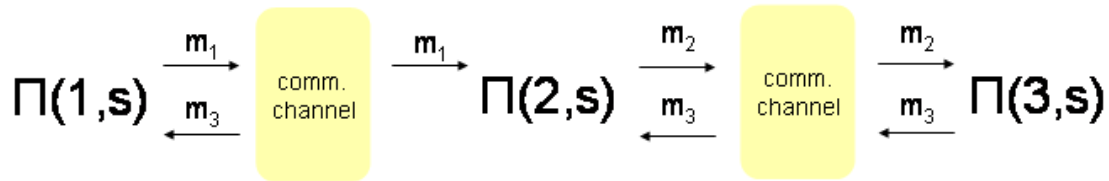
- session IDs are built from the union of partial conversations

$\text{sid}(i,s)$	sid_{i_1}	sid_{i_2}	sid_{i_3}
$\text{sid}(1,s)$	-	m_1	m_3
$\text{sid}(2,s)$	m_1	-	$m_2 m_3$
$\text{sid}(3,s)$	m_3	$m_2 m_3$	-

Partnering Condition in BCPQ

- $\Pi(i,s)$ and $\Pi(j,s)$ are *directly partnered* if
 - they have accepted and
 - $\text{sid}(i,s) \cap \text{sid}(j,t) \neq \emptyset$
 denoted $\Pi(i,s) \leftrightarrow \Pi(j,t)$
- $\Pi(i,s)$ and $\Pi(j,s)$ are *partnered* if
 - there is a sequence $\Pi(l_1,s), \dots, \Pi(l_t,s)$
 - $\Pi(l_1,s) = \Pi(i,s), \Pi(l_t,s) = \Pi(j,s)$
 - $\Pi(l-1,s) \leftrightarrow \Pi(l,s)$ for all $l=2, \dots, t$

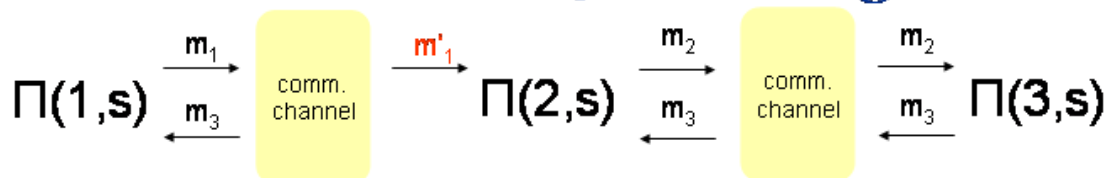
Example of Partnering in BCPQ



$\text{sid}(i,s)$	sid_{i1}	sid_{i2}	sid_{i3}
$\text{sid}(1,s)$	-	m_1	m_3
$\text{sid}(2,s)$	m_1	-	$m_2 m_3$
$\text{sid}(3,s)$	m_3	$m_2 m_3$	-

- $\Pi(1,s) \leftrightarrow \Pi(2,s)$ since $\text{sid}_{12} = \text{sid}_{21}$
- $\Pi(2,s) \leftrightarrow \Pi(3,s)$ since $\text{sid}_{23} = \text{sid}_{32}$
- $\Pi(3,s) \leftrightarrow \Pi(1,s)$ since $\text{sid}_{31} = \text{sid}_{13}$

Drawback of BCPQ Partnering



$\text{sid}(i,s)$	sid_{i1}	sid_{i2}	sid_{i3}
$\text{sid}(1,s)$	-	m_1	m_3
$\text{sid}(2,s)$	m'_1	-	$m_2 m_3$
$\text{sid}(3,s)$	m_3	$m_2 m_3$	-

- $\Pi(1,s) \not\leftrightarrow \Pi(2,s)$ since $\text{sid}_{12} \neq \text{sid}_{21}$, but
- $\Pi(1,s) \leftrightarrow \Pi(3,s)$ and $\Pi(3,s) \leftrightarrow \Pi(2,s)$, thus
- $\Pi(1,s)$ and $\Pi(2,s)$ are still partnered

Insufficient for MA-security!

A More General Partnering Condition

- $\Pi(i,s)$ and $\Pi(j,s)$ are *partnered* if
 - they have accepted
 - $\text{sid}(i,s) = \text{sid}(j,s)$
 - $\text{pid}(i,s) = \text{pid}(j,s)$ where $\text{pid}(i,s)$ consists of all $U_j \in \mathbf{U}$ participating in the session
- used e.g. in [KS05] for UC-secure GKE where $\text{sid}(i,s)$ is given as input
- we assume, either $\text{sid}(i,s)$ is given as input or it is defined in the GKE protocol

Adv. Queries to $\Pi(i,s)$ ^[BCPQ01,KY03,BCP02]

Passive/Active Attacks

- $\text{Send}(\Pi(i,s), m)$ sends m to $\Pi(i,s)$

Known-Key Attacks

- $\text{RevealKey}(\Pi(i,s))$ returns $k(i,s)$, if accepted

Corruptions

- next slide

Weak vs. Strong Corruptions

Current Separation in GKE

- **weak corruption model**^[BCPQ01, KY03]
 - access to $\text{Corrupt}(U_i)$ which reveals LL_i
- **strong corruption model**^[BCP02]
 - access to $\text{Corrupt}(\Pi(i,s))$ which reveals LL_i and $\text{state}(i,s)$
 - **impossible to reveal $\text{state}(i,s)$ without LL_i**

Separation by [CK01] for 2-Party-KE

- **[CK01] specifies two queries**
 - $\text{Corrupt}(U_i)$ reveals LL_i
 - $\text{RevealState}(\Pi(i,s))$ reveals $\text{state}(i,s)$

Current Separation in 2-Party-KE

- **weak corruption model**
 - access to $\text{Corrupt}(U_i)$ but not to $\text{RevealState}(\Pi(i,s))$
- **strong corruption model**
 - access to $\text{Corrupt}(U_i)$ and to $\text{RevealState}(\Pi(i,s))$
 - **possible to ask $\text{RevealState}(\Pi(i,s))$ w/o $\text{Corrupt}(U_i)$**

Instance Freshness wrt. Strong Corr.

- technical core of AKE-Security
- adversary tries to distinguish keys accepted by fresh instances from some random values
- $\Pi(i,s)$ is *fresh* if
 - it has accepted with some $k(i,s)$
 - no `RevealKey()` to $\Pi(i,s)$ or to any partner $\Pi(j,s)$to model (strong) forward secrecy^[BCPQ01,KY03,(BCP02)]
 - no `Corrupt($\Pi(i,s)$)` before `Send($\Pi(j,s), m$)`

Our Refinements: Corruption Models

- based on separation of queries^[CK01]
- in addition to `Send` and `RevealKey` queries

	weak corruption model (wcm)	wcm forward secrecy (wcm-fs)	wcm backward secrecy (wcm-bs)	strong corruption model (scm)
<code>Corrupt(U_i)</code>	no	yes	no	yes
<code>RevealState($\Pi(j,s)$)</code>	no	no	yes	yes

- wcm equiv. to secrecy w/o corruptions
- wcm-fs equiv. to weak corruptions^[BCPQ01,KY03,CK01]
- scm equiv. to strong corruptions^[BCP02,CK01]

Our Refinements:

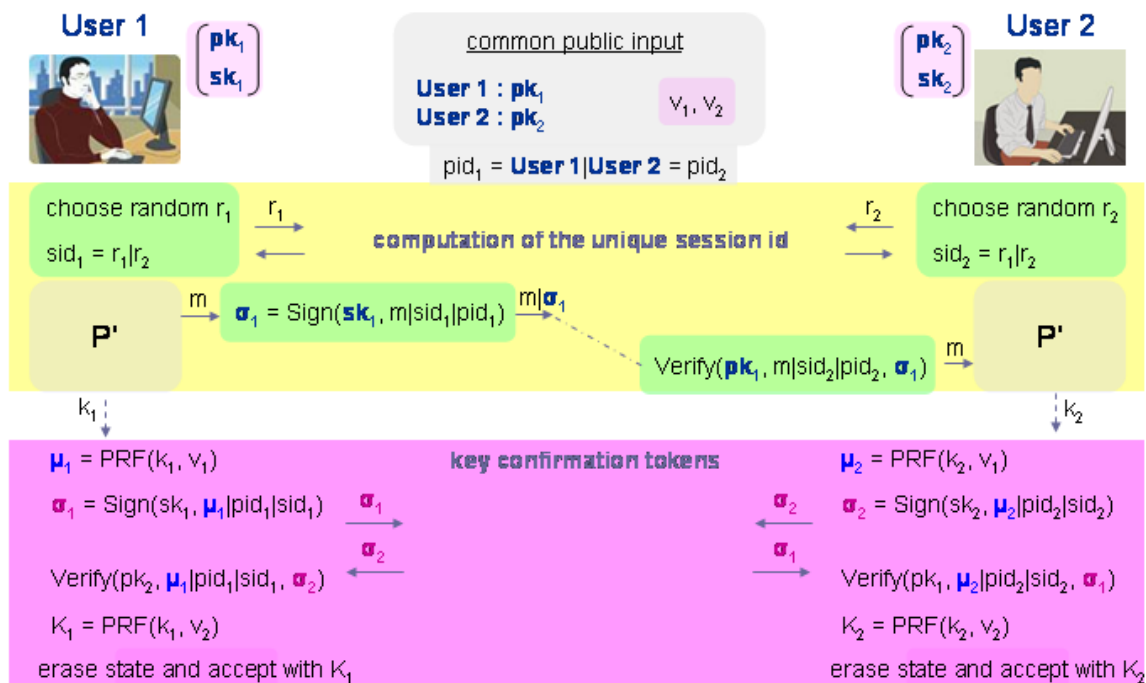
Secrecy Types / Instance Freshness

- **wfs** equiv. to forward secrecy^[BCPQ01,KY03]
 - no $\text{Corrupt}(\Pi(i,s))$ before $\text{Send}(\Pi(j,s), m)$
- **sfs** equiv. to strong forward secrecy^[BCP02]
 - as in wfs and no RevealState to $\Pi(i,s)$ or its partners before they accept
- **wbs** (weak backward secrecy)
 - no RevealState to $\Pi(i,s)$ or its partners after protocol invocation
- **sbs** (strong backward secrecy) *(of theoretical value)*
 - as in wbs and no $\text{Corrupt}(\Pi(i,s))$ before $\text{Send}(\Pi(j,s), m)$ after protocol invocation

Security Compiler C_{AMA}

[KY03]

+ [KS05]



Security of C_{AMA}

- we show that C_{AMA} preserves AKE-Security for
 - weak/strong forward secrecy, weak backward secrecy
 - original [KY03] considered only weak forward secrecy
 - [KY03] provides stronger security than originally proven, but
 - original [KY03] is not generic (more on this in the paper)
- provides MA-Security for
 - scm
 - implies security of [KS05] even if the adversary reveals state($\Pi(i,s)$) for instances of uncorrupted U_i
 - possible due to the separation of queries
 - [KS05] provides stronger security than originally proven

Summary of Contributions

- identify weaknesses of partnering in [BCPQ01]
- suggest more general treatment of partnering
- adapt separation of Corrupt and RevealState^[CK01] to GKE
- refine AKE with weak/strong backward/forward secrecy
- redefine MA while allowing RevealState to uncorrupted users (not in the talk)
- define C_{AMA} as mod. combination of [KY03] and [KS05]
- show why original [KY03] is not generic (not in the talk)
- prove that C_{AMA} satisfies our AKE and MA (not in the talk)