Part II
Bellare-Rogaway Model (Active Adversaries)
Active Attacks

Adversary may tamper, drop, or inject messages in executions
Identities
Identities?

In the passive security model, both scenarios are identical from server's view. Therefore, need identities to distinguish good and bad cases in active model.
Identities!

- Certified $pk_C$ (via $cert_C$)
- Certified $pk_S$ (via $cert_S$)
- $sk_C$
- $sk_S$

Both parties also output intended partner identity $pid$

Warning: We do not consider revocation nor registering adversarial keys here!
Implications for Security Model

Users are assigned user id uid

Each party with identity uid receives \((pk_{uid}, sk_{uid}, cert_{uid})\)

Adversary may recover \(sk_{uid}\) from \(pk_{uid}\)
Adding Corruption

\[
\{pk_{\text{uid}}\} \quad \xrightarrow{\text{uid}_C, \text{uid}_S} \quad (\text{transcript}_{\text{id}}, \text{id}) \\
\quad \xrightarrow{\text{id}} \quad K_b \\
\quad \xrightarrow{\text{id}} \quad K_{\text{id}} \\
\quad \xrightarrow{\text{uid}} \quad \text{sk}_{\text{uid}}
\]

\[
\text{EXEC} \quad \text{pick id} \\
\text{TEST}_b \quad \text{key } K_{\text{id}} \\
\text{REVEAL} \quad \text{secret random bit } b: \quad \text{return } K_0 = K_{\text{id}} \text{ (if } b=0) \text{ or } K_1 = $ \text{ (if } b=1)
\]

\[
\text{transcript}_{\text{id}} \quad \text{key } K_{\text{id}}
\]
New Attack Surfaces

certified $pk_C$ (via $cert_C$)

1. Corrupt client to learn $sk_C$
2. Impersonate client to derive Key K
3. TEST server key

(intended partner is C)
Attacks via false Identities

not via corruption, but through rogue certificates

Indian government agency issues Google certificates

Some systems trusted the fake certificates, some didn’t, moved quickly to tell others to revoke them.

By Larry Seltzer for Zero Day | July 9, 2014 -- 13:07 GMT (14:07 BST) | Topic: Security

French gov used fake Google certificate to read its workers' traffic

Liberté, égalité ... invisibilité: Homme-dans-l'intermédiaire espionnant et trésorier de...

Rogue web certificate could have been used to attack Iran dissidents

Flaw could have let attackers steal passwords and data from apparently secure connections to Google sites such as Gmail
Extensions: Corruption

State

Adversary learns sk_C but also state (randomness,...)?

(“weak“ vs. „strong“ corruption)

Complete take-over

Can client still run executions after corruption?

Here: Adversary only gets sk_C and corrupt party can still be active
Authenticating the Partner

Anonymous

Unilateral

Mutual

intended partner is S

pk_s

pk_c

intended partner is S

intended partner is C
Sessions
Conceptual Change: Sessions

Passive adversaries: honest parties run execution

Active adversaries: unclear if there is partner at all
Adding SEND

\[ \{pk_{uid}\} \rightarrow (id, msg) \rightarrow \text{next-msg} \rightarrow \text{SEND} \]

also: initiate session id

\[ \text{id} \rightarrow K_b \rightarrow \text{TEST}_b \]

\[ \text{id} \rightarrow K_{id} \rightarrow \text{REVEAL} \]

\[ \text{uid} \rightarrow sk_{uid} \rightarrow \text{CORRUPT} \]

session key \( K_{id} \)
Replacing EXEC with SEND

Warning: for forward secrecy later it is advantageous to also use EXEC
Freshness Condition?

Adversary should not be allowed to TEST one party and REVEAL other party in the following scenario:

Active but somewhat passive attack: Client and Server derive same key

need a notion that sessions belong together
### Session Matching or Partnering

<table>
<thead>
<tr>
<th>Bellare-Rogaway (BR93)</th>
<th>Matching conversations</th>
<th>Crypto `93</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bellare-Rogaway (BR95)</td>
<td>Partnering Function</td>
<td>STOC `95</td>
</tr>
<tr>
<td>Bellare-Pointcheval-Rogaway (BPR00)</td>
<td>Session identifiers</td>
<td>Eurocrypt 2000</td>
</tr>
</tbody>
</table>
Matching Conversations

Sessions are partnered if identical transcripts and in chronological order:

Sometimes defined without chronological order:
Partnering Functions

Uses notion of
(not necessarily efficiently computable)
partnering function $f: \{\text{transcripts}\} \rightarrow \{\text{id}\}$

Sessions are partnered if
$f(\text{transcript}) = f(\text{transcript}^\ast)$

Not used anywhere anymore
Session Identifiers

specify session identifier sid

Sessions are partnered if
sid = sid‘

sid usually defined through (partial) transcript
Restrictions Apply

1. Session identifiers should be unique:
   \[ \text{Prob[ three honest parties with same sid ]} \approx 0 \]

2. Same sid in genuine execution between two honest parties

3. Same sid, same key
Uniqueness is not hard

Common example: TLS
Freshness

**Mutual Authentication**
- neither TEST session nor partner session REVEALED
- neither party in TEST nor intended partner pid CORRUPT

**Unilateral Authentication**
- ...
  - +
  - if unauthenticated partner then there is honest partner session

**Anonymous**
- ...
  - +
  - there is honest partner session
Authenticated Key Exchange

Adversary wins if $a=b$ and freshness condition satisfied

KE is BR-secure against active adversaries if for any efficient adversary: $\Pr[A\ \text{wins}] \leq 1/2 + \text{neg}$
At most one other party (≤1) holds the session key
(and for authenticated cases,
if intended partner is honest then it is that party)

Do you see why it cannot be three parties?

Key confirmation (≥1):
Another party holds the key

see also: Fischlin, Günther, Schmidt, Warinschi: Key Confirmation in Key Exchange…, S&P 2016
Teaser for the Break

We have defined security for single TEST query:

\[ \text{id} \xrightarrow{K_b} \text{TEST}_b \]

Is it equivalent if adversary has multiple TEST queries?

\[ \text{id} \xrightarrow{K_b} \text{TEST}_b \]

Hint: consider first how you need to change the TEST oracle and then how you could ensure this in a reduction to the single-query case.