Compiler for group key exchange

Speaker: Yangguang Tian

December 13, 2014

Reference

1. Mark Manulis, Provably Secure Group Key Exchange, 2007
2. Jonathan Katz, Moti Yung, Scalable Protocols for Authenticated Group Key Exchange, 2007
3. Mike Burmester, Yvo Desmedt, A secure and Efficient Conference Key Distribution System, 1994

Outline

- Basic definition of authenticated key exchange (AKE), mutual authentication (MA) and contributiveness (Con).
- Efficient Burmester-Desmedt protocol and compiler for AKE-security.
- Concrete proof of compiler for AKE-security.
- Conclusion

Compiler

Definition (Mark Manulis): A security-enhancing Group Key Exchange (GKE) protocol compiler $\mathcal{C}$ is a protocol which takes as input a GKE protocol $\mathcal{P}$ and outputs a compiled GKE protocol $\mathcal{P}'$ with security properties not provided by $\mathcal{P}$.

$$\mathcal{P} \rightarrow \mathcal{C} \rightarrow \mathcal{P}'$$

Generic security-enhancing solutions!

Compilers

- Any secured key exchange protocols can be transferred to **authenticated** key exchange (AKE) via compiler for AKE-security;
- Any secured authenticated key exchange protocols can be transferred to authenticated key exchange with **mutual authentication** (MA) via compiler for MA-security;
- Any secured authenticated key exchange protocols with MA can be transferred to AKE+MA protocol with **contributiveness** (Con) via compiler for n-contributiveness;
- Multi-purpose compiler (any combination of them), which is depending to protocol requirement.

UNIVERSITY OF WOLLONGONG

KY-Model

- $\prod_i^{s_i}$ indicates instance s_i of user i.
- Execute, Send (active adversary), (Session key) Reveal, Corrupt, Test query (fresh session).
- PID, SID, Partnering, Freshness* (1, no reveal query; 2, no send query **after** corrupt query).
- Comparison with CK model. 2 party↔group party.

Authenticated Key Exchange

- Aim: From passive to active in view of adversary.
- Definition: Outsider (active) adversary $\mathcal{A}$ wins if she could distinguish real session key and random value.

Mutual Authentication

- Aim: subsume unknown key share resistant, key confirmation.
- Definition: $\prod_i^{s_i}$ accepted session key $k_i^{s_i}$, and is known to malicious participant (insider). $\mathcal{A}$ wins game if:
  - These is **no** instance oracle $\prod_j^{s_j}$, that shared identical PID, SID,

    or
  - There is **an** instance oracle $\prod_j^{s_j}$ accepted $k_j^{s_j} \neq k_i^{s_i}$, that shared identical PID, SID.
- Two conditions are corresponding to?

Contributiveness

- Aim: subsume key control, unpredictability.
- Definition: $\mathcal{A}$ wins game if **all** holds:
  - $\prod_i^{s_i}$ in terminate, accept pre-defined session key $k_i^{s_i}$, no corrupt query;
  - corrupt at most $n-1$ users who are belonging to PID.
- Informally, $\mathcal{A}$ wins if there exist at least one honest user who accept the session key that chosen previously by adversary.

UNIVERSITY OF WOLLONGONG

Key Exchange Protocol: Efficient BD protocol

- Star-based, Tree-based, Broadcast-based, Ring-based.
- Round 1: Each user broadcasts $R_i = g^{r_i}$ to others;
- Round 2: Each user computes $X_i = (\frac{R_{i+1}}{R_{i-1}})^{r_i} = \frac{g^{r_{i+1} * r_i}}{g^{r_{i-1} * r_i}}$;
- Key computation:

$$K_i = R_{i-1}^{n * r_i} * X_i^{n-1} * X_{i+1}^{n-2} ... * X_{i-2}$$
$$= g^{r_1 * r_2 + r_2 * r_3 ... r_{n-1} * r_n}$$

- Analysis: Lacking of authentication!

UNIVERSITY OF WOLLONGONG

Compiler for AKE-security

- Round 1: Each user broadcasts $(0, r_i, ID_i)$ to others; $PID = (ID_1||...ID_n)$, $SID = (r_1||...||r_n)$
- Round 2: Each user broadcasts $(1, R_i, \sigma_i)$ to others, where $\sigma_i = Sig_{x_i}(1, R_i, PID, SID)$;
- Round 3: After verification, each user broadcasts $(2, X_i, \sigma'_i)$ to others, where $\sigma'_i = Sig_{x_i}(2, X_i, PID, SID)$;
- Key computation: After verification, each user computes K_i as before.
- Analysis: Purpose of sequence number, nonce? Authentication via? How to formally proof it? From KE to AKE!

UNIVERSITY OF WOLLONGONG

Preliminary

- Game-hopping technique, why use it? What is the end point of game?
- Simulation changed slightly! not jump too far.
- Transition based on failure events, condition events, bridge, indistinguishability.
  Difference Lemma:

$$|P[A] - P[B]| \leq P[E]$$
$$\Leftrightarrow P[A \wedge \overline{E}] = P[B \wedge \overline{E}]$$

- Concrete Example: From secured KE protocol (BD) to AKE. BD as building block.

Game 0

- G_0 : This is **real** game between adversary $\mathcal{A}$ and simulator $\mathcal{S}$, $\mathcal{S}$ has to answer all queries made by $\mathcal{A}$ under specific (e.g., **CK**) model. We denote Win_i^{AKE} as the probability of $b' = b$ at respective games.

Game 1

- G_1 : This game is identical to game G_0 except that $\mathcal{S}$ will fail and set b' as random if a nonce r_i is used by an uncorrupted instance oracle in two different sessions. We define this event as Repeat:

$$\left|Pr[Win_0^{AKE}] - Pr[Win_1^{AKE}]\right| \leq Pr[Repeat] \leq n * m^2/2^k \qquad (1)$$

- Purpose: Prevent replay attack, guarantee session identifier is unique.

Game 2

- G_2 : This game is identical to game G_1 except that $\mathcal{S}$ will fail and set b' as random if $\mathcal{A}$'s send query in form of (σ_i, m_i), where σ_i is **valid** signature that **not** previously generated by simulator **before** issuing corrupt query to an uncorrupted party. We define this event as Forge.

$$\left|Pr[Win_1^{AKE}] - Pr[Win_2^{AKE}]\right| \leq n * Pr[Forge] \qquad (2)$$

- Purpose: No forgery attack, since **secured** digital signature used for authentication, as a building block.

Sub-summary

- From active to passive in term of adversary's attacking capability. It actually **removed** active adversary's replay and forgery ability, which in turn equals to passive adversary.
- It paves the way for following games.

Game 3

- G_3 : $\mathcal{S}$ will set **g-th** session as target session. $\mathcal{S}$ will fail and set b' as random if $\mathcal{A}$ issue test query **not** occur in the **g-th** session.
- We denote this event as Guess. $Pr[Guess] = 1/m$

$$\begin{aligned} Pr[Win_3^{AKE}] &= Pr[Win_3^{AKE} \wedge Guess] + Pr[Win_3^{AKE} \wedge \overline{Guess}] \\ &= Pr[Win_2^{AKE}] * 1/m + 1/2 * (1 - 1/m) \end{aligned}$$

Game 4

- G_4 : In this game, we consider simulator $\mathcal{S}$ no longer acts as just an simulator, but an (passive) attacker against KE protocol, denote it as $\mathcal{A}_{KE}$. However, $\mathcal{A}_{KE}$ in this game might **not** completely simulate all queries made by active $\mathcal{A}$ against AKE since his attacking capability only confined to KE protocol. $\mathcal{A}_{KE}$ does **additional** computations based on specification of AKE.
- $\mathcal{A}_{KE}$ has access to activate, (send), corrupt, session-key reveal, test oracles under specified model (**weaker** than CK model, but attacking capability is **identical** to active $\mathcal{A}$).
- $\mathcal{A}_{KE}$ will get real session key K from test oracle.

Game 4

$$\mathcal{A}\frac{queries}{answers}\mathcal{A}_{KE}(Simulator) \Leftrightarrow \mathcal{O}_i(Different\ \ oracles)$$

Particularlly, based on queries from $\mathcal{A}$, $\mathcal{S}_{KE}$ answers either directly from oracles $\mathcal{O}_i$, or constructs it under specification of AKE.
When $\mathcal{A}$ issues test query (it is corresponding to g-th session), then $\mathcal{A}_{KE}$ will

$$Response \leftarrow \begin{cases} SK \leftarrow K & b = 1 \\ R \in_R \{0,1\}^k & b = 0 \end{cases}$$

$$Pr[Win_3^{AKE}] = Pr[Win_4^{AKE}] \tag{3}$$

Game 5

- G_5 : In this game, $\mathcal{A}$ will get random value R (instead of K) from test oracle.

$$\mathcal{A} \frac{test\ query}{SK/R} \mathcal{A}_{KE}$$

$$Response \leftarrow \begin{cases} SK \leftarrow R & b = 1 \\ R \in_R \{0,1\}^k & b = 0 \end{cases}$$

$$\left| Pr[Win_4^{AKE}] - Pr[Win_5^{AKE}] \right| \leq Adv_{\mathcal{A}}^{KE} \tag{4}$$

- Obviously, $\mathcal{A}$ will get nothing information about b **except** random guess.

$$Pr[Win_5^{AKE}] = 1/2. \tag{5}$$

UNIVERSITY OF WOLLONGONG

Remarks

- After compiler for AKE-security, we can continue to use compiler for MA-security (Katz-Shin), or compiler for contributiveness.
- Each game's transition?
- Game 3-5 can reduced to one game! Transition based on indistinguishability.

$$SK \leftarrow \begin{cases} K & G_2 \\ R & G_3 \end{cases}$$

- No matter how many building blocks are being used, we always can reduce the proposed protocol to those building blocks. That is why using game-hopping technique.

UNIVERSITY OF WOLLONGONG

Conclusion

- Compiler is depending on security requirement.
- Proposed protocol can be reduced to building blocks.
- All mentioned compilers are generic, when applied to concrete GKE protocols, we need to consider further efficiency optimization.

Thanks for your time!
Question?

UNIVERSITY OF WOLLONGONG