PAKE in the UC-Framework

Adaptive Security

David Pointcheval
CNRS-ENS-INRIA
Paris - France

CANS '2007
Singapore
Sunday, December 9th, 2007

Universal Composability
Password-Based AKE
UC Password-Based AKE
Security proofs give the guarantee that an assumption is enough for security: if an adversary can break the system, one can break the assumption \( \Rightarrow \) "reductionist" proof.

Proof by Reduction

Reduction of a problem \( P \) to an attack \( Atk \):

Let \( A \) be an adversary that breaks the scheme

Then \( A \) can be used to solve \( P \)

\( A \)
Proof by Reduction

Let $A$ be an adversary that breaks the scheme. Then $A$ can be used to solve $P$.

The instance $I$ of $P$ is intractable $\Rightarrow$ the scheme is unbreakable.

Provably Secure Scheme

To prove the security of a cryptographic scheme, one has to make precise the algorithmic assumptions and the security notions to be guaranteed. A reduction: an adversary can help to break the assumption.
In such a reduction, our simulator tries to emulate the environment, until the adversary may win the attack game. What about the composition of multiple protocols? As soon as an adversary may break one part of the global system, whereas other parts may provide a protection, other executing protocols may provide additional information to the adversary. Therefore, either we re-prove the global system, or we prove each component in the UC Framework.
Real vs. Ideal

Definition of security

Protocol $\pi$ emulates the ideal process for $F$ if for any adversary $A$ there exists a simulator $S$ such that for all $Z$:

$$IDEAL_{F} S, Z \sim EXEC_{\pi}, A, Z.$$ 

$$\Rightarrow$$ we say that protocol $\pi$ securely realizes $F$.

Equivalently:

$$\exists S \exists d \forall Z IDEAL_{F} S, Z \sim EXEC_{\pi}, A, d, Z.$$ 

$$\forall A \forall Z \exists S IDEAL_{F} S, Z \sim EXEC_{\pi}, A, Z.$$
PAKE in the UC-Framework - 13
David Pointcheval

**UC Theorem: Idea**

1. $P_1$
2. $P_2$
3. $P_3$
4. $P_4$

**Ideal Functionality $F$**

$Q_1$, $Q_3$, $Q_4$, $Q_2$

Implications of UC:

- Can design and analyze protocols in a modular way:
  - Partition a given task $T$ to simpler sub-tasks $T_1, \ldots, T_k$.
  - Construct protocols for realizing $T_1, \ldots, T_k$.
  - Construct a protocol for $T$ assuming ideal access to $T_1, \ldots, T_k$.
  - Use the composition theorem to obtain a protocol for $T$ from scratch.
  - (Now can be done concurrently and in parallel.)
Key Exchange: a two-party protocol to generate a common random key that is "secret" for external adversaries.

Assuming authenticated communication (Diffie-Hellman model)
Unauthenticated communication (AKE)

Different ways to authenticate the exchange:
- Long-term public keys for signature or encryption plus "public-key infrastructure".
- Long-term pre-shared keys
- Trusted third parties (The Kerberos model)
- Passwords
PAKE in the UC-Framework

AKE has been studied extensively: Protocols were proposed, and later broken. First complexity-based notion: [Bellare-Rogaway - Crypto '93] Based on a "distinguishing game" for the adversary (FtG). Explicitly handles multiple concurrent sessions. Treatments that argue usability for secure sessions: Bellare-Canetti-Krawczyk - STOC '98 simulation based (but has problems). Canetti-Krawczyk – EC '01: based on BR93 with a different system model, defines and obtains "secure sessions". Canetti-Krawczyk – EC '02: A UC treatment of AKE.

Ideal Functionality: KE

Functionality $F_{KE}$

$F_{KE}$ is parameterized by a security parameter $k$. It interacts with an adversary $S$ and a set of (dummy) parties via the following queries:

Upon receiving a query (NewSession, $sid$, $P_i$, $P_j$, role) from party $P_i$:

Send (NewSession, $sid$, $P_i$, $P_j$, role) to $S$. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record ($P_j$, $P_i$), then record ($P_i$, $P_j$).

Upon receiving a query (NewKey, $sid$, $P_i$, $sk$) from $S$, where $|sk| = k$:

If there is a record ($P_i$, $P_j$), and this is the first NewKey query for $P_i$, then:
- If either $P_i$ or $P_j$ is corrupted, then output ($sid$, $sk$) to player $P_i$.
- If there is also a record ($P_j$, $P_i$), and a key $sk'$ was sent to $P_j$, output ($sid$, $sk'$) to $P_i$.
- In any other case, pick a new random key $sk'$ of length $k$ and send ($sid$, $sk'$) to $P_i$.

Figure 1: The authenticated key-exchange functionality $F_{KE}$
Functionality $F_{\text{pwKE}}$

The functionality $F_{\text{pwKE}}$ is parameterized by a security parameter $k$. It interacts with an adversary $S$ and a set of parties via the following queries:

Upon receiving a query (NewSession, $sid$, $P_1$, $P_2$, $pw$, role) from party $P_i$:
Send (NewSession, $sid$, $P_1$, $P_2$, role) to $S$. In addition, if this is the first NewSession query, or if this is the second NewSession query and there is a record ($P_2$, $P_4$, $pw'$), then record ($P_1$, $P_3$, $pw$) and mark this record fresh.

Upon receiving a query (TestPwd, $sid$, $P_4$, $pw'$) from the adversary $S$:

on receiving a query (NewKey, $sid$, $P_4$, $sk$) from $S$, where $|sk| = k$:
If there is a record of the form ($P_1$, $P_3$, $pw$), and this is the first NewKey query for $P_4$, then:

- If this record is compromised, or either $P_1$ or $P_3$ is corrupted, then output ($sid$, $sk$) to player $P_i$.
- If this record is fresh, and there is a record ($P_1$, $P_3$, $pw'$) with $pw' = pw$, and a key $sk'$ was sent to $P_3$, and ($P_1$, $P_4$, $pw$) was fresh at the time, then output ($sid$, $sk'$) to $P_i$.
- In any other case, pick a new random key $sk'$ of length $k$ and send ($sid$, $sk'$) to $P_i$.

Either way, mark the record ($P_1$, $P_4$, $pw$) as completed.

Figure 2: The password-based key-exchange functionality $F_{\text{pwKE}}$
In this ideal functionality:
TestPwd query, which gives the adversary to test one password per session.
In case of correct password guess, the adversary can choose the key.

Passwords:
The environment chooses the passwords. Can thus make players run with different passwords, or related passwords. ⇒ passwords are not in an internal state of the functionality: no need of joint-state UC.

### Concurrent Executions

KOY/GL Protocol

2 &; &; 7
Commitment:
\[ c = \text{Commit}(pw, r) = \text{Encrypt}(\text{pke}, pw, r) \]

IND-CCA \(\Rightarrow\) NM for multiple commitments

Smooth Projective Hash Functions:
\[ H(c, pw) = \text{Hash}(hk; c, pw) = \text{ProjHash}(hp; c, pw, r) \]

No information about \(H(c, pw)\) if \(pw \neq \text{Decrypt}(\text{ske}, c)\)

Hard to compute \(H(c, pw')\) without either the hash-key \(hk\) or the witness \(r\)

Session Key:
\[ c_1 = \text{Encrypt}(\text{pke}, pw, r_1) \quad c_2 = \text{Encrypt}(\text{pke}, pw, r_2) \]
\[ \text{sk} = \text{Hash}(hk_2; c_1, pw) + \text{ProjHash}(hp_1; c_2, pw, r_2) = \text{ProjHash}(hp_2; c_1, pw, r_1) + \text{Hash}(hk_1; c_2, pw) \]

Passive Adversary:
- Pseudo-randomness without the witness \(\Rightarrow\) indistinguishability of the session key

Active Adversary:
- NM for multiple commitments \(\Rightarrow\) no new valid commitment (except chance with \(pw\))
- Invalid commitment \(\Rightarrow\) indistinguishability of \(\text{sk}\) (statistic)
- Replay of commitment: does not know the witness \(\Rightarrow\) indistinguishability of \(\text{sk}\) (computational)
Proof: with an extractable commitment

Adversary sends $c_1$: we can extract the password, and check whether it is correct or not

Simulator sends $c_1$: with a random/dummy pw!

Wrong simulation if adversary has guessed pw

Not negligible and thus not UC secure

UC Password-Based AKE

Universal Composability

Password-Based AKE

UC Password-Based AKE
PAKE in the UC-Framework - 27

David Pointcheval

UC PAKE

Canetti-Halevi-Katz-Lindell-MacKenzie – EC '05

PAKE in the UC-Framework - 28

David Pointcheval

CHKLMK: Idea

UC Proof: with an extractable commitment

Adversary sends $c_0$: we can extract the password, and check whether it is correct or not

Simulator sends $c_0$: with a random/dummy password!

adversary sends $c_1$: extract and check

Wrong $\Rightarrow$ random key

correct $\Rightarrow$ we commit the correct password in $c_2$ and simulate a fake ZKP

\[
\begin{align*}
c_0 & \leftarrow E_{pke}(pw; r_0) \\
c_0 & \rightarrow c_0 \\
c_0, vk & \leftarrow \text{sigKey}() \\
c_1 & \leftarrow E_{pke}(pw; r_1) \\
c_1, c_2, hp & \rightarrow (sk, vk) \\
c_2 & \leftarrow E_{pke}(pw, r_2) \\
hk & \leftarrow H \\
hp & \leftarrow \alpha(hk; c_1) \\
c_2, hp & \rightarrow c_2, hp \\
ZKP(c_0 \approx c_2) & \leftarrow \text{sign}_{sk}(c_0, hp, hp') \\
hp', \sigma & \leftarrow \text{sign}_{sk}(c_0, hp, hp') \\
os & \leftarrow \text{sign}_{sk}(c_2, hp, hp') \\
\end{align*}
\]

\[
\begin{align*}
\text{session-key} & \leftarrow \Pi_{hk}(c_1, pw) \\
& \quad + H_{hk}(c_2, pw; r_2) \\
\text{session-key} & \leftarrow h_{hp}(c_1, pw, r_1) \\
& \quad + H_{hk}(c_2, pw; r_2)
\end{align*}
\]
An adaptive adversary can corrupt players at any time and receive the internal state in a KOY/GL-like scheme: not secure in the simulation, use of "dummy password" for $c_0$ if corruption right after that: how to simulate $r_0$?

in an EKE-like scheme: secure granted the Programmability of the Ideal-Cipher and the Random Oracle $\Rightarrow$ Adaptive adversaries and strong corruption

[Abdalla-Catalano-Chevalier-Pointcheval – CT-RSA '08]