Universal Composability

- Password-Based AKE
- UC Password-Based AKE
Provable Security

Security proofs give the guarantee that an assumption is enough for security:
- if an adversary can break the system
- one can break the assumption
⇒ “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 $P$
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 $P$

Instance $I$ of $P$ → Solution of $I$

$P$ intractable $\Rightarrow$ scheme unbreakable

Provably Secure Scheme

To prove the security of a cryptographic scheme, one has to make precise

- the algorithmic assumptions
- the security notions to be guaranteed
- a reduction: an adversary can help to break the assumption
Simulation

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?
- the simulation fails 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.

Either we re-prove the global system, or we prove each component in the UC Framework.

Universal Composability

[Canetti - FOCS '01]

Ideal process:

Protocol execution:

Protocol \( \pi \) securely realizes \( F \) if:
- For any adversary \( A \)
- There exists a simulator \( S \)
- Such that no environment \( Z \) can tell whether it interacts with:
  - a run of \( \pi \) with \( A \)
  - an ideal run with \( F \) and \( S \)
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 $Z$

$$\text{IDEAL}_{F,S,Z}^F \sim \text{EXEC}_{\pi,A,Z}.$$ 

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

$$\left( \forall A \right) \left( \exists S \right) \left( \forall Z \right) \text{IDEAL}_{F,S,Z}^F \sim \text{EXEC}_{\pi,A,Z}.$$ 

Equivalently:

$$\left( \exists S_d \right) \left( \forall Z \right) \text{IDEAL}_{F,S,Z}^F \sim \text{EXEC}_{\pi,A_d,Z}$$

$$\left( \forall A \right) \left( \forall Z \right) \left( \exists S \right) \text{IDEAL}_{F,S,Z}^F \sim \text{EXEC}_{\pi,A,Z}.$$ 

UC Theorem: Composition

Modular composition
UC Theorem: Idea

\[ A_d \]

\[ P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \]

\[ Q_1 \rightarrow Q_2 \rightarrow Q_3 \rightarrow Q_4 \]

David Pointcheval
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

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 - 17

David Pointcheval

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

---

PAKE in the UC-Framework - 18

Ideal Functionality: KE

---

**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 $F_{KE}$
Password-Based Authentication

- **Asymmetric**: \((sk_A, pk_A)\) and possibly \((sk_B, pk_B)\)
  - they authenticate to each other using the knowledge of the private key associated to the certified public key
- **Symmetric**: common (long – high-entropy) secret
  - they use the long term secret to derive a secure and authenticated ephemeral key \(sk\)
- **Password**: common (short - low-entropy) secret
  - let us assume a 20-bit password
  - \(\Rightarrow\) it is possible to win with non-negligible advantage

---

**Ideal Functionality: pwKE**


The functionality \(\mathcal{F}_{pwKE}\) 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, \(sid, P_i, P_j, pw, role\)) 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 \((P_j, P_i, pw')\), then record \((P_i, P_j, pw)\) and mark this record fresh.

**Upon receiving a query (TestPwd, \(sid, P_i, pw'\)) from the adversary \(S\):**
- If there is a record of the form \((P_i, P_j, pw)\) which is fresh, then do: If \(pw = pw'\), mark the record compromised and reply to \(S\) with “correct guess”. If \(pw \neq pw'\), mark the record interrupted and reply with “wrong guess”.

**Upon receiving a query (NewKey, \(sid, P_i, sk\)) from \(S\), where \(|sk| = k\):**
- If there is a record of the form \((P_i, P_j, pw)\), and this is the first NewKey query for \(P_i\), then:
  - If this record is compromised, or either \(P_i\) or \(P_j\) is corrupted, then output \((sid, sk)\) to player \(P_i\).
  - If this record is fresh, and there is a record \((P_j, P_i, pw')\) with \(pw' = pw\), and a key \(sk'\) was sent to \(P_j\), and \((P_j, P_i, pw)\) 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 \(P_j\).
- Either way, mark the record \((P_i, P_j, pw)\) as completed.

**Figure 2**: The password-based key-exchange functionality \(\mathcal{F}_{pwKE}\)
In this ideal functionality:

- **TestPwd** query, which gives the authorization to 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

### KOY/GL Protocol

**Client** ($P_i$):

1. Choose $c_2 := E_{pke}(pw, r_2)$
2. Compute $hk := H$
3. Compute $hp := \alpha(hk; c_1)$
4. Send $c_1, vk$ to the server

**Server** ($P_s$):

1. Compute $(sk, vk) \rightarrow \text{sigKey}(\$)$
2. Compute $c_1 := E_{pke}(pw, r_1)$
3. Compute $c_2, hp$
4. Send $c_2, hp$ to the client

**Client** ($P_i$):

1. Compute $hk' := H$
2. Compute $hp' := \alpha(hk'; c_2)$
3. Compute $\sigma := \text{Sign}_{sk}(c_2, hp, hp')$
4. If $\text{Verify}_{vk}((c_2, hp, hp'), \sigma) = 1$
   - Compute session-key $\leftarrow H_{hk}(c_1, pw) + h_{hp'}(c_2, pw; r_2)$

**Server** ($P_s$):

1. Compute session-key $\leftarrow h_{hp}(c_1, pw; r_1) + H_{hk'}(c_2, pw)$
Commitment:
- \( c = \text{Commit}(pw, r) = \text{Encrypt}(pke, pw, r) \)
- 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 \( pw \neq \text{Decrypt}(ske, c) \)
- Hard to compute \( H(c, pw') \) without either the hash-key \( hk \) or the witness \( r \)

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

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 \( sk \) (statistic)
- Replay of commitment: does not know the witness \( \Rightarrow \) indistinguishability of \( sk \) (computational)
KOY/GL: Security Analysis

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 $pw$!
  - adversary sends $c_2$: extract and check
    - wrong $\Rightarrow$ random key
    - correct $\Rightarrow$ we get stuck

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
**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 $pw$!
  - adversary sends $c_1$: extract and check $pw$
    - wrong $\Rightarrow$ random key
    - correct $\Rightarrow$ we commit the correct password in $c_2$ and simulate a fake ZKP
An adaptive adversary can corrupt players at any time and receive the internal state:

- in 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 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]

---

### EKE Scheme

\[
\begin{align*}
\text{Client } U & : & x \leftarrow [1; q-1] \\
& & y \leftarrow [1; q-1] \\
& & (U1) \; X \leftarrow g^x \\
& & \text{U} \rightarrow \text{S}, X \\
& & \text{(S2)} \; Y \leftarrow g^y \\
& & \text{S} \rightarrow \text{U}, Y^* \\
& & \text{Y}^* \leftarrow \varepsilon_{pw}(Y) \\
& & K_S \leftarrow X^y \\
& & (U3) \; Y = \mathcal{D}_{pw}(Y^*) \\
& & K_U \leftarrow Y^* \\
& & \text{Auth} \leftarrow \mathcal{H}_1(\text{ssid}||U||S||X||Y||K_U) \\
& & sk_U \leftarrow \mathcal{H}_0(\text{ssid}||U||S||X||Y||K_U) \\
& & \text{completed} \\
& & \text{Auth} \leftarrow \mathcal{H}_1(\text{ssid}||U||S||X||Y||K_S) \\
& & (S4) \; \text{if } (\text{Auth} = \mathcal{H}_1(\text{ssid}||U||S||X||Y||K_S)) \\
& & \text{then completed} \\
& & sk_S \leftarrow \mathcal{H}_0(\text{ssid}||U||S||X||Y||K_S) \\
& & \text{else error}
\end{align*}
\]