Two parties (Alice and Bob) agree on a *common* secret key $SK$, in order to establish a secret channel.

- Basic security requirement:
  
  *implicit authentication*

  *only* the intended partners can compute the session key.
**Authentication**

To prevent active attacks, some kind of authentication of the flows is required:

- **Asymmetric**: $\langle sk_A, pk_A \rangle$ and possibly $\langle sk_B, pk_B \rangle$
- **Symmetric**: common (high-entropy) secret
- **Password**: common (low-entropy) secret
  
e.g. a 20-bit password

**Password-based Authentication**

Password (low-entropy secret) *e.g. 20 bits*

- exhaustive search is possible
- basic attack: **on-line exhaustive search**
  
  - the adversary guesses a password
  - tries to play the protocol with this guess
  - failure $\Rightarrow$ it erases the password from the list
  - and restarts...
  
  - after 1,000,000 attempts, the adversary wins
  
  cannot be avoided

We want it to be the **best attack**…

**Dictionary Attack**

- **Off-line exhaustive search**
  
  - a few passive or active attacks
  - failure/transcript $\Rightarrow$ erasure of **MANY** passwords from the list: this is called dictionary attack

- To prevent them:
  
  - a passive eavesdropping
    
    - no *useful* information about the password
  - an active trial
    
    - cancels *at most one* password

**Summary**

- Password-based Authenticated Key Exchange
  
  - EKE, OKE and a generalization
  
  - Trapdoor Hard-to-Invert Isomorphisms

- Examples
Efficiently Samplable

- $f_{pk}$ must be *trapdoor* “hard-to-invert”, not necessarily “one-way”: but just *samplable*
  
  \[(r, c) \leftarrow S(pk) \text{ such that } r \text{ random in } M_{pk} \text{ and } c = f_{pk}(r)\]

- $pk$ must be easy to generate
- $f_{pk}$ must be a bijection ⇒ *to be checked*

Hard-to-Invert: not Enough?

When $pk$ is chosen by Alice

- $sk$ is unknown to the adversary
  
  - the adversary can know only one pre-image $r$ (for the guessed password $\pi$)
  - for other $\pi$'s, the “hard-to-invert” property prevents from extracting/checking other $r$ values

This is the intuition... For the formal proof

- Hard-to-invert
- Bijection
- Morphism

Morphism: for the Proof

For checking a password, one uses $k$ or $SK$

- one must compute $r$ (appears in $H$-$H'$ queries)
- Either $c'$ sent by Bob: from any correct $(\pi, r)$ such that $c' = f_{pk}(r) \otimes G(\pi)$, one can invert $f_{pk}$
  - by simulating $c' = f_{pk}(a)$ for a known $a$
  - by embedding the challenge $y$ in $G(\pi)$
    
    \[y = c' \otimes f_{pk}(a) = f_{pk}(r) \otimes f_{pk}(a) = f_{pk}(r-a)\]

- Or by the adversary: from two correct pairs $(\pi, r)$

Trapdoor Hard-to-Invert Isomorphisms Family

\[F = (f_{pk})_{pk}\] trapdoor hard-to-invert isomorphisms

- $(pk, sk) \leftarrow G(1^k)$: generation
  
  - $f_{pk}$ is an isomorphism from $M_{pk}$ onto $G_{pk}$
- $(r, c) \leftarrow S(pk)$: sample
  
  - such that $r$ random in $M_{pk}$ and $c = f_{pk}(r)$ (random in $G_{pk}$)
- Given $y$ and $pk$, check whether $y \in f_{pk}(M_{pk}) = G_{pk}$
- Given $y$ and $sk$, easy to invert $f_{pk}$ on $y$
- Without $sk$, hard to invert $f_{pk}$
Summary

- Password-based Authenticated Key Exchange
- EKE, OKE and a generalization
  - Trapdoor Hard-to-Invert Isomorphisms

Candidates

**Diffie-Hellman**: $sk = x$, $pk = g^x$

$$f_{pk}(g^a) = g^{ax} = pk^a$$

$g_{sk}(b) = b^{1/x}$

$f_{pk}$ is not one-way, but hard-to-invert
under the **CDH assumption**

⇒ classical DH-AKE variants (PAK or AuthA)

**RSA**: $sk = d$, $pk = (n,e)$

$f_{pk}$ is one-way under the **RSA assumption**,
buts $pk$ must contain a valid RSA key: NIZK proof

⇒ variant of “protected OKE”

Candidates (Cont'd)

**Square root**: $sk = (p,q)$, $pk = n$

$f_{pk}$ is an automorphism onto $\text{QR}_n^*$,
but for specific moduli only (Blum moduli)

⇒ to be checked: can be done (verified) efficiently

$f_{pk}$ is one-way under
the **integer factoring problem**

⇒ the first **Password-Based Authenticated Key Exchange** based on **factoring**