Password-based Authenticated Key Exchange
State of the Art

David Pointcheval
CNRS-ENS – France

LBNL
Berkeley – California - USA
August 2004

Summary

- Authenticated Key Exchange
- Password-based Authentication
- Encrypted Key Exchange
- Open Key Exchange
- Implementation Concerns
- In Practice

Summary

- ▶ Authenticated Key Exchange
- Password-based Authentication
- Encrypted Key Exchange
- Open Key Exchange
- Implementation Concerns
- In Practice

Authenticated Key Exchange

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

- Intuitive goal: ***implicit authentication***
  - *only* the intended partners can compute the session key
- Formally: ***semantic security***
  - the session key SK is indistinguishable from a random string RS, to anybody else

Additional Properties

- ***Mutual authentication***
  - They are both sure to *actually* share the secret with the people they think they do
- ***Forward-secrecy***
  - Even if a long-term secret data is corrupted (leaked to the adversary), previously shared secrets are *still* semantically secure

---

The Leakage of Information

- The protocol is run over a public network, then the transcripts are public:
  - an **execute**-query provides such a transcript to the adversary
- The secret data SK may be misused (with a weak encryption scheme, ...):
  - the **reveal**-query is answered by this secret data SK

Passive/Active Adversaries

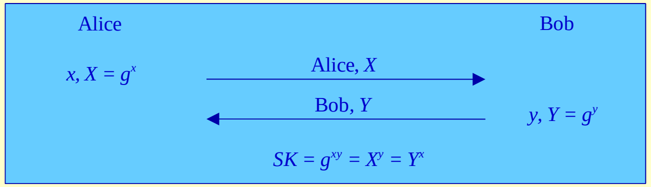
- ***Passive***: history built using
  - **execute**-queries → transcripts
  - **reveal**-queries → session keys
- ***Active***: entire control of the network
  - **send**-queries

    active, adaptive adversary
    on concurrent executions
    - to send message to Alice or Bob (in place of Bob or Alice respectively)
    - to intercept, forward and/or modify messages

Diffie-Hellman Key Exchange

- The most famous key exchange protocol:

Diffie-Hellman



It is ***not** authenticated*: anybody can say
“I am Alice” or “I am Bob”

⇒ semantic security
against ***passive** adversaries*

Authentication

To prevent active attacks *(manufactured **send**)*, some kind of authentication is required:

- **Asymmetric**: (sk_A, pk_A) and possibly (sk_B, pk_B)

  parties authenticate to each other using the knowledge of the private key associated to the certified public key
- **Symmetric**: common (high-entropy) secret

  they use the long term secret to derive a secure and authenticated ephemeral key SK
- **Password**: common (low-entropy) secret

  e.g. a 20-bit password

Summary

- Authenticated Key Exchange
- Password-based Authentication
- Encrypted Key Exchange
- Open Key Exchange
- Implementation Concerns
- In Practice

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 ⇒ it erases the password from the list
  - and restarts…
- after 1,000,000 attempts, the adversary wins

cannot be avoided

Dictionary Attack

- **On-line exhaustive search**
  - cannot be avoided
  - can be made less serious (delay, limitations, …)

  We want it to be the **best attack**…

- **Off-line exhaustive search**
  - a few passive or active attacks
  - failure ⇒ erasure of **MANY** passwords from the list

  this is called **dictionary attack**

Security

One wants to prevent dictionary attacks:

- any passive trial (**execute** + **reveal**)
  - no *useful* information about the password
- one active trial (**send**)
  - cancels *at most one* password
    from the list of possible passwords

Summary

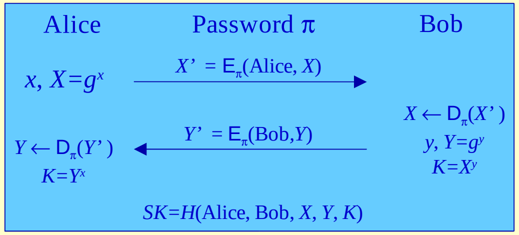
- Authenticated Key Exchange
- Password-based Authentication
- Encrypted Key Exchange
- Open Key Exchange
- Implementation Concerns
- In Practice

EKE = Encrypted Key Exchange

EKE = Diffie-Hellman, with encrypted flows

- Must be done carefully…
- From X' ,
  for any password π
  - decrypt X'
  - check whether
    it begins with “Alice”

bad one



Alice Password π Bob

$x, X=g^x$

$X' = E_\pi(\text{Alice}, X)$ →

$X \leftarrow D_\pi(X')$
$y, Y=g^y$
$K=X^y$

← $Y' = E_\pi(\text{Bob}, Y)$

$Y \leftarrow D_\pi(Y')$
$K=Y^x$

$SK=H(\text{Alice}, \text{Bob}, X, Y, K)$

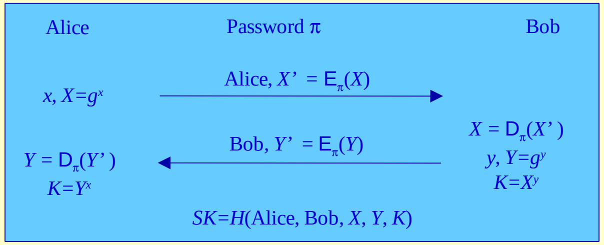
avoid any redundancy

EKE = Encrypted Key Exchange

Bellovin-Merritt 1992

The correct scheme:

without redundancy



Alice Password π Bob

$x, X=g^x$

$\text{Alice}, X' = E_\pi(X)$ →

$X = D_\pi(X')$
$y, Y=g^y$
$K=X^y$

← $\text{Bob}, Y' = E_\pi(Y)$

$Y = D_\pi(Y')$
$K=Y^x$

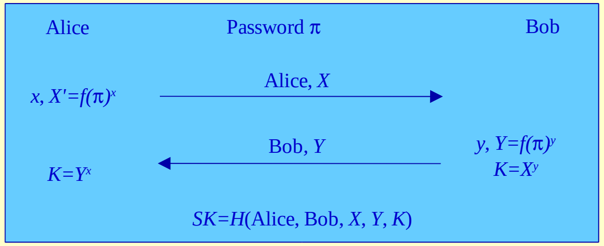
$SK=H(\text{Alice}, \text{Bob}, X, Y, K)$

- E_π **must** be a ***bijection*** from the group $<g>$

SPEKE = Simple Password Exponential Key Exchange

Jablon 1996

Variant of DH-EKE:



Alice Password π Bob

$x, X'=f(\pi)^x$ $\xrightarrow{\text{Alice}, X}$

$\xleftarrow{\text{Bob}, Y}$ $y, Y=f(\pi)^y$

$K=Y^x$ $K=X^y$

$SK=H(\text{Alice}, \text{Bob}, X, Y, K)$

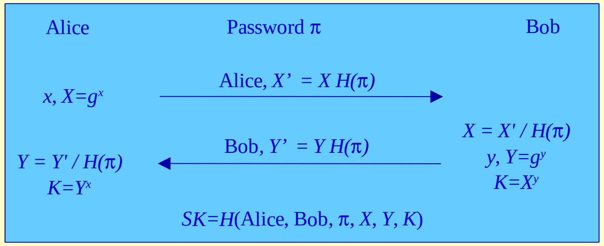
- According to the function *f*, this scheme can be either secure or totally insecure
  - If *f* is a random function (random oracle) onto the ***whole*** group $<g>$: **provably secure**

[MacKenzie 2001]

PPK – AuthA - MDHKE

Boyko-MacKenzie-Patel 2000 - Bellare-Rogaway 2000
Bresson-Chevassut-P. 2003/2004

A simple variant: ***one-time pad*** $E_\pi(X) = X\,H(\pi)$



Alice Password π Bob

$x, X=g^x$ $\xrightarrow{\text{Alice}, X' = X\,H(\pi)}$

$\xleftarrow{\text{Bob}, Y' = Y\,H(\pi)}$ $X = X'/H(\pi)$, $y, Y=g^y$

$Y = Y'/H(\pi)$, $K=Y^x$ $K=X^y$

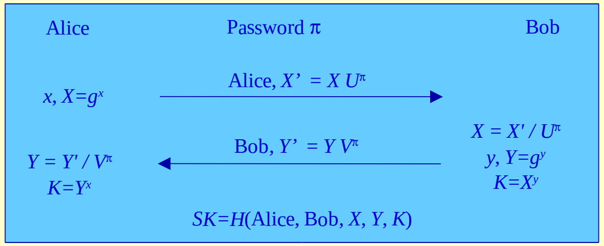
$SK=H(\text{Alice}, \text{Bob}, \pi, X, Y, K)$

- If *H* is a random function (random oracle) onto the ***whole*** group $<g>$: **provably secure**

Simple Encrypted Key Exchange

Abdalla-P. 2004

The simplest variant: $E_\pi(X) = X\,U^\pi$



Alice Password π Bob

$x, X=g^x$ $\xrightarrow{\text{Alice}, X' = X\,U^\pi}$

$\xleftarrow{\text{Bob}, Y' = Y\,V^\pi}$ $X = X'/U^\pi$, $y, Y=g^y$

$Y = Y'/V^\pi$, $K=Y^x$ $K=X^y$

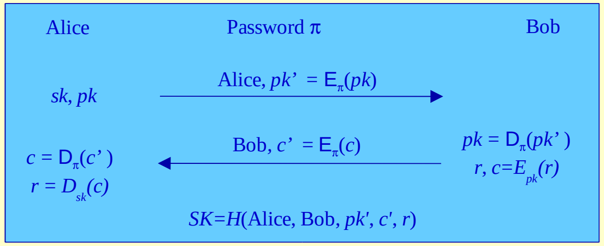
$SK=H(\text{Alice}, \text{Bob}, X, Y, K)$

- No random function onto groups
- Just two fixed elements *U*, *V* in $<g>$
- Non-concurrent executions: **provably secure**

Generalized Encrypted Key Exchange

- More generally
  - Alice generates a public key *pk*, sends *pk* ***encrypted with the password*** → *pk'*
  - Bob recovers *pk*, generates a random *r*, encrypts it with *pk* → *c*, sends *c* ***encrypted with the password*** → *c'*
  - Alice recovers the common random *r*
  - The session key *SK* is derived from this *r*

Generalized Encrypted Key Exchange



Alice　Password π　Bob

sk, pk

Alice, $pk' = E_\pi(pk)$ →

$pk = D_\pi(pk')$
$r, c = E_{pk}(r)$

← Bob, $c' = E_\pi(c)$

$c = D_\pi(c')$
$r = D_{sk}(c)$

$SK = H(\text{Alice}, \text{Bob}, pk', c', r)$

- Problems:
  - pk must be truly random (no redundancy)
  - pk and r are not on the same space, in general
- Nice exception: ElGamal (DH-EKE)
  - requires E_π to be over $<g>$
  - impossible to be used with RSA…

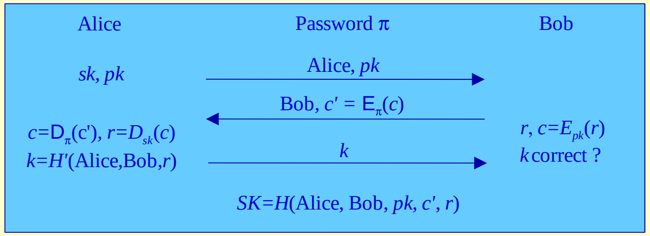
Summary

- Authenticated Key Exchange
- Password-based Authentication
- Encrypted Key Exchange
- ▶ Open Key Exchange
- Implementation Concerns
- In Practice

Open Key Exchange

Lucks 1997

- The public key pk is sent in **clear**:



Alice　Password π　Bob

sk, pk

Alice, pk →

← Bob, $c' = E_\pi(c)$

$r, c = E_{pk}(r)$

$c = D_\pi(c'), r = D_{sk}(c)$
$k = H'(\text{Alice}, \text{Bob}, r)$

k →

k correct ?

$SK = H(\text{Alice}, \text{Bob}, pk, c', r)$

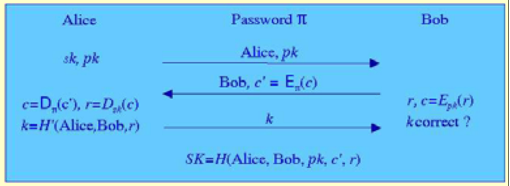
- Requirements to avoid partition attacks:
  - E_π must be a bijection from the ciphertext space
  - E_{pk} must be a surjection onto this space

Surjection

Since pk can be chosen by the adversary,
one must check "**E_{pk} is a surjection**"

- If not, given c', one eliminates the π's, that lead to c's which are not in the image set of E_{pk}: *partition attack*
- If so, given c', any π is possible: sending the correct k means *guessing the good* π

⇒ zero-knowledge proof
  - concurrent
  - non-malleable



Alice　Password π　Bob

sk, pk

Alice, pk →

← Bob, $c' = E_\pi(c)$

$r, c = E_{pk}(r)$

$c = D_\pi(c'), r = D_{sk}(c)$
$k = H'(\text{Alice}, \text{Bob}, r)$

k →

k correct ?

$SK = H(\text{Alice}, \text{Bob}, pk, c', r)$

Protected OKE

Lucks 1997

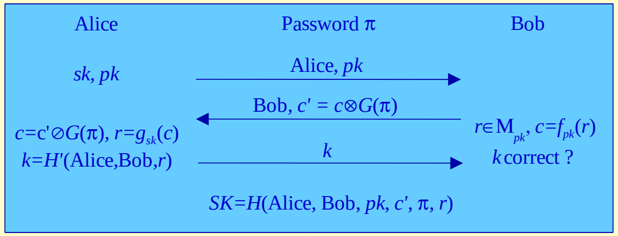
- Specific to RSA
- Additional proof of valid modulus
  - Flaw in the original scheme
  - Repaired in SNAPI
    [MacKenzie-Patel-Swaminathan – 2000]
    - But not very efficient:
      very large exponents ($e > n$)
  - Efficient variant with RSA-IPAKE
    [Catalano-P. -Pornin - 2004]

IPAKE = Generalized OKE

Catalano-P. -Pornin 2004

- Using any isomorphism and one-time pad

Isomorphism for Password-based Authenticated Key Exchange



Alice | Password π | Bob

sk, pk $\xrightarrow{\text{Alice}, pk}$

$\xleftarrow{\text{Bob}, c' = c \otimes G(\pi)}$ $r \in \mathrm{M}_{pk}, c = f_{pk}(r)$

$c = c' \oslash G(\pi), r = g_{sk}(c)$
$k = H'(\text{Alice}, \text{Bob}, r)$ $\xrightarrow{k}$ k correct ?

$SK = H(\text{Alice}, \text{Bob}, pk, c', \pi, r)$

  - f_{pk} isomorphism from F_{pk} onto the group $(\mathrm{G}_{pk}, \otimes)$
  - Must be *trapdoor* “hard-to-invert”

IPAKE = Applications

- ElGamal encryption (Diffie-Hellman function)
  - PAK [Boyko-MacKenzie-Patel 2000]
  - AuthA [Bellare-Rogaway 2000]
  - OMDHKE [Bresson-Chevassut-P. 2003]
- RSA function
  - Protected OKE [Lucks 1997]
  - SNAPI [MacKenzie-Patel-Swaminathan 2000]
- Modular square
  - The first Password-based AKE
    related to ***integer factoring***

Summary

- Authenticated Key Exchange
- Password-based Authentication
- Encrypted Key Exchange
- Open Key Exchange
- ▶ Implementation Concerns
- In Practice

Several Candidates

Many proposals:

- In the **standard model**: KOY protocol (*DDH*)

  [Katz-Ostrovsky-Yung 2001]

  - But quite inefficient
- In the **random oracle model**:
  - EKE (*CDH*), OKE (*CDH/RSA*), SPEKE (*CDH*)
  - IPAKE (*CDH/RSA/Fact*), simple EKE (*CDH*)

Which seem very efficient…

Full-domain Functions

Most of these schemes require full-domain random functions/bijections:

- Hash function onto $<g>$
- Block cipher over $<g>$

- How to implement them?
  - Take a hash function / block cipher onto $\{0,1\}^k$, where k is the length of any encoding of elements in $<g>$
  - Iterate it until falling in $<g>$

Implementation Details

Requirement:

- 2^k / Card($<g>$), must not be too large
  - Average number of iterations

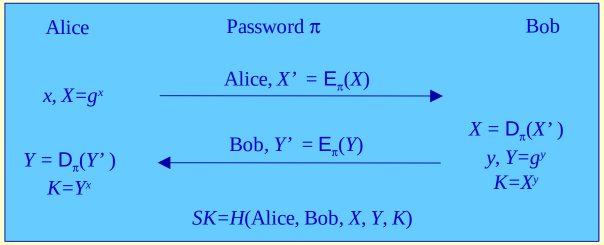
$\Rightarrow$ g of (almost) maximal order

$\Rightarrow$ use of large exponents… or ***elliptic curves***

- In $\mathbf{Z}_p^*$, $|p| = 1024 \Rightarrow$ exponents over >1000 bits
  - Small subgroups are possible,
    but at the same high cost (large co-factor)
- On EC, 160 bit field $\Rightarrow$ exponents over 160 bits
  - Just one iteration on average

Timing Attacks

The number of iterations may depend on the password:



Alice — Password π — Bob

$x, X=g^x$ $\xrightarrow{\text{Alice}, X' = \mathsf{E}_\pi(X)}$ $X = \mathsf{D}_\pi(X')$, $y, Y=g^y$, $K=X^y$

$Y = \mathsf{D}_\pi(Y')$, $K=Y^x$ $\xleftarrow{\text{Bob}, Y' = \mathsf{E}_\pi(Y)}$

$SK=H(\text{Alice}, \text{Bob}, X, Y, K)$

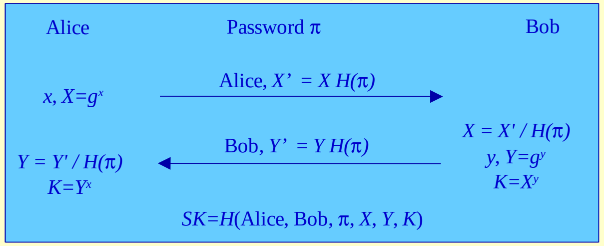
On basic EKE: responding time = number of iterations for (X, X') and (Y, Y')

- Each passive attack divides the set by 4

$\Rightarrow$ **partition attack!**

One-Time Pad

The use of the one-time pad limits the damages:



Alice — Password π — Bob

$x, X=g^x$ $\xrightarrow{\text{Alice}, X' = X\,H(\pi)}$

$X = X' / H(\pi)$
$y, Y=g^y$
$K=X^y$

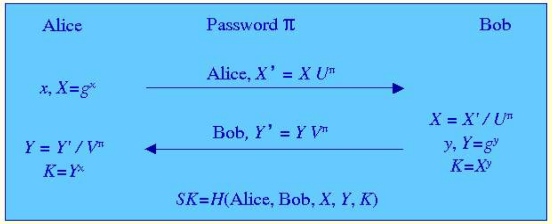
$\xleftarrow{\text{Bob}, Y' = Y\,H(\pi)}$ $Y = Y' / H(\pi)$
$K=Y^x$

$SK=H(\text{Alice}, \text{Bob}, \pi, X, Y, K)$

- No pre-computation: responding time = number of iterations for $H(\pi)$
  - But always the same information
- Pre-computation of $H(\pi)$: no information leaked

Simple EKE

- Particular case: the “simple EKE”



Alice — Password π — Bob

$x, X=g^x$ $\xrightarrow{\text{Alice}, X' = X\,U^\pi}$

$X = X' / U^\pi$
$y, Y=g^y$
$K=X^y$

$\xleftarrow{\text{Bob}, Y' = Y\,V^\pi}$ $Y = Y' / V^\pi$
$K=Y^x$

$SK=H(\text{Alice}, \text{Bob}, X, Y, K)$

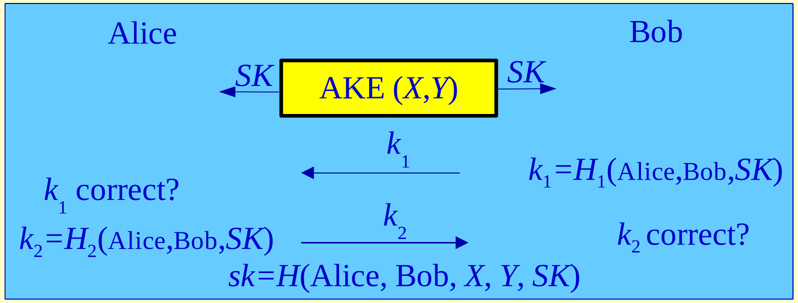
- No full-domain function: easy to implement
- Apply to any prime sub-group: quite efficient
- But: restricted to ***non-concurrent*** executions

Summary

- Authenticated Key Exchange
- Password-based Authentication
- Encrypted Key Exchange
- Open Key Exchange
- Implementation Concerns
- ▶ In Practice

Additional Properties

- Mutual authentication:
  - General construction (with key confirmation) [Bellare-P. -Rogaway 2000]



Alice — Bob

$SK \leftarrow$ AKE (X,Y) $\rightarrow SK$

$k_1 = H_1(\text{Alice},\text{Bob},SK)$ $\xleftarrow{k_1}$ k_1 correct?

$k_2 = H_2(\text{Alice},\text{Bob},SK)$ $\xrightarrow{k_2}$ k_2 correct?

$sk = H(\text{Alice}, \text{Bob}, X, Y, SK)$

- Forward secrecy:
  - EKE/OKE provide it [Abdalla-Chevassut-P. 2004]

Hostile Environments

- The client machine may not be fully trusted:
  - When the user types his password: leaked…
  - ⇒ for some schemes, it is possible to use any kind of ephemeral secret shared between the user and the server (OTP, SecurID)

Work in progress…

Vulnerable Server

- The server machine may not be fully trusted:
  - The machine may be vulnerable, in case of corruption, all the passwords are leaked…
  - ⇒ verifier-based variants exist (the server just owns a verifier –an image of the password throught a one-way function)
    - In case of corruption, a dictionary attack is necessary
    - By adding salts, it is made less effective
  - ⇒ the password can be distributed among several servers (threshold AKE)

Work in progress…