Augmented Encrypted Key Exchange

Shalier Xia

Encrypted Key Exchange

- Two parties $A$ and $B$ agree on a shared password ($P$), base ($\alpha$), and modulus ($\beta$)
- $A$ chooses a random number $R_A$ and encrypts $\alpha^{R_A} \pmod{\beta}$ and sends to $B$
- $B$ also chooses a random number $R_B$ and encrypts $\alpha^{R_B} \pmod{\beta}$
- $B$ then uses the shared password to decrypt $P[\alpha^{R_A} \pmod{\beta}]$ and calculates $(\alpha^{R_A R_B}) \pmod{\beta}$.
  - The session key, $K$, is derived from this

New Notation: $K[info]$ – symmetric encryption of “info” with key $K$
- $B$ uses the $K$ to encrypt a random challenge and sends to $A$: $P[\alpha^{R_B} \pmod{\beta}], K[challenge_B]$
Encrypted Key Exchange

• A then uses the shared password to decrypt $P[\alpha^{Rn} \pmod{\beta}]$ and calculates the key, K

• A uses K to decrypt $K[\text{challenge}_B]$ and generates a random challenge$_A$ and appends challenge$_B$ then encrypts it, giving us $K[\text{challenge}_A, \text{challenge}_B]$ which is transmitted to B

• B decrypts $K[\text{challenge}_A, \text{challenge}_B]$ and verifies that challenge$_B$ was appended and sends to A, $K[\text{challenge}_A]$

• A decrypts $K[\text{challenge}_A]$ and verifies that it matches challenge$_A$
Encrypted Key Exchange

Pro:
• Because the session key depends on input from both A and B, defeats man-in-the-middle attacks

Con:
• Requires the server, party B, to store passwords in plaintext
Augmented Encrypted Key Exchange

- $H(P)$ – the password-hashing function
- $F(P,K)$ – a one-way function that depends on the password and session key
- $T(H(P), F(P,K), K)$ – a predicate that evaluates to true iff the original password was used to create $H(P)$ and $F(P,K)$
  - $T(X, Y, Z)$ is true iff $X = H(P')$ and $Y = F(P', Z)$, for some $P'$
Augmented Encrypted Key Exchange

- Two parties $A$ and $B$ agree on a shared encrypted password, $H(P)$, base ($\alpha$), and modulus ($\beta$)

- $A$ chooses a random number $R_A$ and encrypts $\alpha^{R_A} \pmod{\beta}$, with $H(P)$ and sends to $B$

- $B$ also chooses a random number $R_B$ and encrypts $\alpha^{R_B} \pmod{\beta}$

- $B$ then uses the encrypted password $H(P)$ to decrypt $H(P)[\alpha^{R_A} \pmod{\beta}]$ and calculates $(\alpha^{R_A R_B}) \pmod{\beta}$.
  - The session key, $K$, is derived from this

- $B$ uses the $K$ to encrypt a random challenge and sends to $A$:

$$H(P)[\alpha^{R_B} \pmod{\beta}], K[\text{challenge}_B]$$
Augmented Encrypted Key Exchange

• A then uses the encrypted password to decrypt $H(P)[\alpha^{RB} \pmod{\beta}]$ and calculates the key, $K$

• A uses $K$ to decrypt $K[\text{challenge}_B]$ and generates a random challenge $\text{challenge}_A$ and appends $\text{challenge}_B$ then encrypts it, giving us $K[\text{challenge}_A, \text{challenge}_B]$ which is transmitted to $B$

• $B$ decrypts $K[\text{challenge}_A, \text{challenge}_B]$ and verifies that $\text{challenge}_B$ was appended and sends to $A$, $K[\text{challenge}_A]$

• $A$ decrypts $K[\text{challenge}_A]$ and verifies that it matches $\text{challenge}_A$
Augmented Encrypted Key Exchange

• What if the adversary is able to obtain $H(P)$?
  • Then the adversary could mimic the two parties
• So, we extend the protocol so that $A$ sends $K[F(P,K)]$ to $B$
• $B$ decrypts $K[F(P,K)]$ and uses the predicate $T(H(P), \tilde{F}(P, K), K)$
  • if the predicate evaluates to true then the protocol is successful.

• If the adversary obtains $H(P)$ it can still mimic $B$ to $A$, but without knowing $P$ it cannot mimic $A$ to $B$. 
A-EKE Using Digital Signature Scheme

• Let $V_P$ be the public signature key
• To compute $F(P,K)$, $A$ must sign $K$ with the private key, which is generated from $P$
• $V_P[X]$ – Uses value of key from signature system as key to a symmetric encryption
• $S_k(M)$ – Digital signature of $M$ with private key $S_k$
• $V_k(X,M)$ – Verification of signature $X$ of message $M$ with public key $V_k$
• $K^{-1}[info]$ — Symmetric decryption of “info” with key $K$
Brief Review on Digital Signatures:

• Key generation algorithm

• Signing algorithm
  • takes private key and message and outputs signature

• Verification Algorithm
  • takes public key, message and potential signature and outputs a bit
A-EKE Using Digital Signature Scheme

• Two parties A and B agree on a public signature key $V_p$, base ($\alpha$), and modulus ($\beta$)
• A chooses a random number $R_A$ and encrypts $\alpha^{R_A} \pmod{\beta}$ and sends to B
• B also chooses a random number $R_B$ and encrypts $\alpha^{R_B} \pmod{\beta}$
• B then uses the shared public signature key to decrypt $V_p[\alpha^{R_A} \pmod{\beta}]$ and calculates $(\alpha^{R_A} R_B) \pmod{\beta}$.
  • The session key, $K$, is derived from this
• B uses the K to encrypt a random challenge and sends to A:
  $V_p[\alpha^{R_B} \pmod{\beta}], K[\text{challenge}_B]$
A-EKE Using Digital Signature Scheme

- $A$ then uses the shared public signature key to decrypt $P[\alpha^{RB \ (mod \ \beta)]$ and calculates the key, $K$
- $A$ uses $K$ to decrypt $K[\text{challenge}_B]$, and generates a random challenge $\text{challenge}_A$ and appends challenge$_B$ then encrypts it, giving us $K[\text{challenge}_A, \text{challenge}_B]$ which is transmitted to $B$
- $B$ decrypts $K[\text{challenge}_A, \text{challenge}_B]$, verifies that challenge$_B$ was appended and sends to $A$, $K[\text{challenge}_A]$.
- $A$ decrypts $K[\text{challenge}_A]$ and verifies that it matches challenge$_A$
A sends $K[Sp(K)]$ to $B$

B decrypts this with $K$ to obtain $Sp(K)$ and concludes the protocol iff the predicate $Vp(K^{-1}[K[Sp(K)], K]$ is true.

- $V_k(X,M)$ – Verification of signature $X$ of message $M$ with public key $V_k$
- $S_k(M)$ – Digital signature of $M$ with private key $S_k$
Security Analysis of A-EKE

• Two scenarios:
  • Adversary doesn’t have access to $H(P)$
  • Adversary does have access to $H(P)$
Adversary without H(P)

- A knows the password
- B knows H(P)
- Public knowledge: H( ), F( ), and T( )
- Assumption: arguments for functions are unrelated to P or any legitimate session key
- Let EKE phase be a black box, where both parties can negotiate the random session key, K, without releasing any useful info on H(P) or K
- Then A sends K[F(P,K)] to B which is evaluated with the predicate
- Each session has a new session key so despite the adversary learning \( K_1[F(P,K_1)] \), \( K_2[F(P,K_2)] \)... it’s useless for attacking later protocol executions
Adversary with H(P)

- During the EKE phase, the adversary won’t learn anything new
  - Everything uses H(P) which the adversary already knows
- Then A sends K[F(P,K)] to B which is evaluated with the predicate
- The adversary may know K and so can determine F(P,K) but won’t be able to obtain P as F( ) is a one-way function
- Additionally B ensures the session key is unique every time so despite the adversary knowing F(P,K₁), F(P,K₂)... it will not be sufficient enough to compute the new Kᵢ