One-Round Key Exchange with Strong Security: An Efficient and Generic Construction in the Standard Model

Florian Bergsma  Tibor Jager  Jörg Schwenk

PKC 2015
Public-Key Authenticated Key Exchange

Alice

Insecure channel

Bob
Public-Key Authenticated Key Exchange

Alice
(pk_A, sk_A)

Insecure channel

Bob
(pk_B, sk_B)
Public-Key Authenticated Key Exchange

Alice
(pk_A, sk_A)

Insecure channel

Bob
(pk_B, sk_B)

Key k_{AB}

Key k_{AB}

AKE Protocol
Public-Key Authenticated Key Exchange

Alice
(pk_A, sk_A)

Bob
(pk_B, sk_B)

Insecure channel

AKE Protocol

Key k_{AB}
Partner = Bob

Nobody except for Bob is able to distinguish k_{AB} from random
One-Round Key Exchange (ORKE)

\[(pk_A, sk_A) \quad (pk_B, sk_B)\]

\[m_A = f(pk_B, sk_A, r_A)\]
\[m_B = f(pk_A, sk_B, r_B)\]

\[KDF(pk_B, sk_A, r_A, m_B) = k_{AB} = KDF(pk_A, sk_B, r_B, m_A)\]
One-Round Key Exchange (ORKE)

Possibly sent *simultaneously* (or *precomputed*)

\[
m_A = f(pk_B, sk_A, r_A)
m_B = f(pk_A, sk_B, r_B)
\]

\[
KDF(pk_B, sk_A, r_A, m_B) = k_{AB} = KDF(pk_A, sk_B, r_B, m_A)
\]
One-Round Key Exchange (ORKE)

\[(pk_A, sk_A)\] (possibly sent \textit{simultaneously} (or \textit{precomputed}))

\begin{align*}
m_A &= f(pk_B, sk_A, r_A) \\
m_B &= f(pk_A, sk_B, r_B)
\end{align*}

\[KDF(pk_B, sk_A, r_A, m_B) = k_{AB} = KDF(pk_A, sk_B, r_B, m_A)\]

- **Simple** design and implementation
- **Quick** key establishment in at most one RTT
Security Analysis of AKE Protocols

- Provide A with “execution environment” that formalizes A’s capabilities
Security Analysis of AKE Protocols

- A controls the network

- Provide A with “execution environment” that formalizes A’s capabilities
Security Analysis of AKE Protocols

- Provide A with “execution environment” that formalizes A’s capabilities
Security Analysis of AKE Protocols

- A may corrupt sessions

- Provide A with “execution environment” that formalizes A’s capabilities
Security Analysis of AKE Protocols

A has to distinguish k from random

- Provide A with “execution environment” that formalizes A’s capabilities
Weak Randomness in Practice

Many examples for the difficulty in practice:

• Debian OpenSSL PRNG Bug (2006-2008)
• Weak RSA public keys
  – Lenstra et al. (Crypto 2012)
  – Heninger et al. (USENIX Security 2012)
  – Bernstein et al. (Asiacrypt 2013)
• Cold boot attacks
  – Halderman et al. (USENIX Security 2008)

```c
int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
    // guaranteed to be random.
}
```

https://xkcd.com/221/
"eCK Security"

[LLM07]

Key $k_{AB}$ is indistinguishable even if the attacker learns the randomness.

\[ m_A = f(pk_B, sk_A, r_A) \]
\[ m_B = f(pk_A, sk_B, r_B) \]

\[ g(pk_B, sk_A, r_A, m_B) = k_{AB} = g(pk_A, sk_B, r_B, m_A) \]
Forward Security (PFS)
(Diffie, van Oorschot, Wiener, DESI 1992)
Forward Security (PFS)  
(Diffie, van Oorschot, Wiener, DESI 1992)
Forward Security (PFS)
(Diffie, van Oorschot, Wiener, DESI 1992)

“Corruption of the long-term secret should not compromise sessions that were established before the corruption”

- Put forward by large Internet companies since 2011 (Google)
- Design goal of modern protocols like TLS 1.3, TextSecure, ...
The Difficulty of Forward Security in the eCK Model

Forward security: key-indistinguishability is based on secret ephemeral randomness
The Difficulty of Forward Security in the eCK Model

\[\text{Forward security: } \text{key-indistinguishability is based on secret ephemeral randomness}\]

\[\text{eCK security: } \text{key-indistinguishability even if ephemeral randomness is leaked}\]
The Difficulty of Forward Security in the eCK Model

**Forward security:**
key-indistinguishability is based on secret ephemeral randomness

**eCK security:**
key-indistinguishability even if ephemeral randomness is leaked
The Difficulty of Forward Security in the eCK Model

**Forward security:**
key-indistinguishability is based on secret ephemeral randomness

---

**eCK security:**
key-indistinguishability even if ephemeral randomness is leaked

---

Session keys must depend on both long-term and ephemeral secrets, such that corruption of either (but not both) does not corrupt the security of session keys
Contributions

• eCK-PFS secure key exchange
  – One-round (ORKE)
  – First from \textit{generic assumptions}
    • Signature scheme
    • Pseudorandom function
    • Non-interactive key exchange
    • \textbf{First not based on discrete log} type assumption
  – Without Random Oracles
  – Relatively \textbf{efficient}
  – \textbf{Simple} construction and proof
Non-Interactive Key Exchange (NIKE)
(Diffie, Hellman `76; Freire, Hofheinz, Kiltz, Paterson, PKC `13)

Security parameter $\rightarrow$ \text{Gen}_{\text{nike}} \rightarrow (pk, sk)$

$(sk_A, pk_B) \rightarrow \text{KDF}_{\text{nike}} \rightarrow k_{AB}$
Non-Interactive Key Exchange (NIKE)
(Diffie, Hellman ´76; Freire, Hofheinz, Kiltz, Paterson, PKC ´13)

Security parameter $\xrightarrow{}$ $Gen_{nike}$ $(pk, sk)$

$(sk_A, pk_B) \xrightarrow{}$ $KDF_{nike}$ $k_{AB}$

$(pk_A, sk_A)$ \hspace{5cm} $\nexists$ \hspace{5cm} $(pk_B, sk_B)$

$KDF_{nike}(pk_B, sk_A) = k_{AB} = KDF_{nike}(pk_A, sk_B)$
Our Protocol

\[ pk_A := (pk_{A,\text{sig}}, pk_{A,\text{nike}}) \]

\[ pk_B := (pk_{B,\text{sig}}, pk_{B,\text{nike}}) \]
Our Protocol

\[ pk_A := (pk_A, \text{sig}, pk_A, \text{nike}) \]

\[ pk_B := (pk_B, \text{sig}, pk_B, \text{nike}) \]

\[ (pk'_A, sk'_A) \leftarrow \text{NIKEGen}(1^k, r_A) \]

\[ (pk'_B, sk'_B) \leftarrow \text{NIKEGen}(1^k, r_B) \]

\[ m_A = (pk'_A, \text{sig}_A(pk'_A)) \]

\[ m_B = (pk'_B, \text{sig}_B(pk'_B)) \]

\[ KDF_{orke}(pk_B, sk_A, m_B, r_A) = k_{AB} = KDF_{orke}(pk_A, sk_B, m_A, r_B) \]
Our Protocol

\[ pk_A := (pk_{A,sig}, pk_{A,nike}) \]

\[ pk_B := (pk_{B,sig}, pk_{B,nike}) \]

\[ (pk'_A, sk'_A) \leftarrow \text{NIKEGen}(1^k, r_A) \]

\[ (pk'_B, sk'_B) \leftarrow \text{NIKEGen}(1^k, r_B) \]

\[ m_A = (pk'_A, \text{sig}_A(pk'_A)) \]

\[ m_B = (pk'_B, \text{sig}_B(pk'_B)) \]

\[ \text{KDF}_{\text{orke}}(pk_B, sk_A, m_B, r_A) = k_{AB} = \text{KDF}_{\text{orke}}(pk_A, sk_B, m_A, r_B) \]

Similar to signed Diffie-Hellman, but

- **NIKE** instead of DH
- more complex key derivation
Idea of $KDF_{orke}$

Alice essentially computes:

$$KDF_{orke}(pk_B, sk_A, pk'_B, sk'_A) := KDF_{nike}(pk_B, sk_A) \oplus KDF_{nike}(pk'_B, sk'_A) \oplus KDF_{nike}(pk_B, sk_A) \oplus KDF_{nike}(pk'_B, sk_A)$$
Idea of $\text{KDF}_{\text{orke}}$

Alice essentially computes:

$$\text{KDF}_{\text{orke}}(\text{pk}_B, \text{sk}_A, \text{pk}_B', \text{sk}_A') := \text{KDF}_{\text{nike}}(\text{pk}_B, \text{sk}_A) \oplus \text{KDF}_{\text{nike}}(\text{pk}_B', \text{sk}_A') \oplus \text{KDF}_{\text{nike}}(\text{pk}_B, \text{sk}_A') \oplus \text{KDF}_{\text{nike}}(\text{pk}_B', \text{sk}_A)$$

- Adversary learns Randomness(A) and Randomness(B)
Idea of $\text{KDF}_{\text{orke}}$

Alice essentially computes:

$$\text{KDF}_{\text{orke}}(pk_B, sk_A, pk_B', sk_A') := \text{KDF}_{\text{nike}}(pk_B, sk_A) \oplus \text{KDF}_{\text{nike}}(pk_B', sk_A') \oplus \text{KDF}_{\text{nike}}(pk_B, sk_A') \oplus \text{KDF}_{\text{nike}}(pk_B', sk_A)$$

- Adversary learns Randomness(A) and Randomness(B)
- Adversary learns $\text{SecretKey}(A)$ and $\text{SecretKey}(B)$
Idea of $\text{KDF}_{\text{orke}}$

Alice essentially computes:

$$\text{KDF}_{\text{orke}}(\text{pk}_B,\text{sk}_A,\text{pk}_B',\text{sk}_A') := \text{KDF}_{\text{nike}}(\text{pk}_B,\text{sk}_A) \oplus \text{KDF}_{\text{nike}}(\text{pk}_B',\text{sk}_A') \oplus \text{KDF}_{\text{nike}}(\text{pk}_B,\text{sk}_A') \oplus \text{KDF}_{\text{nike}}(\text{pk}_B',\text{sk}_A)$$

- Adversary learns Randomness(A) and Randomness(B)
- Adversary learns SecretKey(A) and SecretKey(B)
- Adversary learns SecretKey(A) and Randomness(B)
Idea of $KDF_{orke}$

Alice essentially computes:

$$KDF_{orke}(pk_B, sk_A, pk_B', sk_A') := KDF_{nike}(pk_B, sk_A) \oplus KDF_{nike}(pk_B', sk_A') \oplus KDF_{nike}(pk_B, sk_A') \oplus KDF_{nike}(pk_B', sk_A)$$

- Adversary learns Randomness(A) and Randomness(B)
- Adversary learns SecretKey(A) and SecretKey(B)
- Adversary learns SecretKey(A) and Randomness(B)
- Adversary learns Randomness(A) SecretKey(B)
Idea of $KDF_{orke}$

Alice essentially computes:

$$KDF_{orke}(pk_B, sk_A, pk'_B, sk'_A) := KDF_{nike}(pk_B, sk_A) \oplus KDF_{nike}(pk'_B, sk'_A) \oplus KDF_{nike}(pk_B, sk_A') \oplus KDF_{nike}(pk'_B, sk_A')$$

- Adversary learns Randomness(A) and Randomness(B)
- Adversary learns SecretKey(A) and SecretKey(B)
- Adversary learns SecretKey(A) and Randomness(B)
- Adversary learns Randomness(A) SecretKey(B)

Adversary may learn all non-trivial combinations of randomness / long-term secret, even from the “target-session”
The “real” $KDF_{orke}$

Input: $(pk_B, sk_A, (pk_B', \text{sig}_B), (pk_A', \text{sig}_A))$

- $T := \text{sort}((pk_B', \text{sig}_B), (pk_A', \text{sig}_A))$
- $k_1 := \text{PRF}(KDF_{nike}(pk_B, sk_A), T)$
- $k_2 := \text{PRF}(KDF_{nike}(pk_B, sk_A'), T)$
- $k_3 := \text{PRF}(KDF_{nike}(pk_B', sk_A), T)$
- $k_4 := KDF_{nike}(pk_B', sk_A')$
- $k := k_1 \oplus k_2 \oplus k_3 \oplus k_4$

Output $k$
Generic Construction

• Building blocks of the ORKE protocol:
  – Non-interactive key exchange
  – Signature scheme
  – Pseudorandom function

• Instantiable with any concrete construction
  – From different assumptions, like
    • Discrete log type, with/without pairing
    • Factoring-related
    • Possibly post-quantum?
Summary

• eCK-PFS secure construction of ORKE
  – **Simple** and **natural** construction and proof
  – **Generic**, based on **standard primitives**
    • Gives rise to **first ORKE not based on DL**
  – Relatively **efficiently instantiable**
    • Instantiations in ROM: very efficient
    • Instantiations without ROM: not horrible
Summary

• eCK-PFS secure construction of ORKE
  – Simple and natural construction and proof
  – Generic, based on standard primitives
    • Gives rise to first ORKE not based on DL
  – Relatively efficiently instantiable
    • Instantiations in ROM: very efficient
    • Instantiations without ROM: not horrible

Thank you!
## Comparison with other protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Standard Model</th>
<th>PFS</th>
<th>weak PFS</th>
<th>KCI</th>
<th>exp. per party</th>
<th>pairing evaluations</th>
<th>Security model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{T}S1$ [21]</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>1</td>
<td>-</td>
<td>$BR^1$</td>
</tr>
<tr>
<td>$\mathcal{T}S3$ [21]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>3</td>
<td>-</td>
<td>$BR^1$</td>
</tr>
<tr>
<td>MQV</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>1</td>
<td>-</td>
<td>$CK$</td>
</tr>
<tr>
<td>HMQV</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
<td>-</td>
<td>$CK$</td>
</tr>
<tr>
<td>KEA</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>2</td>
<td>-</td>
<td>$CK$</td>
</tr>
<tr>
<td>P1 [6]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>8</td>
<td>2</td>
<td>$CK$</td>
</tr>
<tr>
<td>P2 [6]</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>2</td>
<td>$CK$</td>
</tr>
<tr>
<td>NAXOS</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>4</td>
<td>-</td>
<td>$eCK$</td>
</tr>
<tr>
<td>Okamoto</td>
<td>✓ $+$ $\pi$PRF</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>8</td>
<td>-</td>
<td>$eCK$</td>
</tr>
<tr>
<td>NAXOS$_{pfs}^2$</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4</td>
<td>-</td>
<td>$eCK$-PFS</td>
</tr>
<tr>
<td>ORKE$^3$ (NIKE)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>5</td>
<td>-</td>
<td>$eCK$-PFS</td>
</tr>
<tr>
<td>ORKE$^4$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>16</td>
<td>12</td>
<td>$eCK$-PFS</td>
</tr>
</tbody>
</table>