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

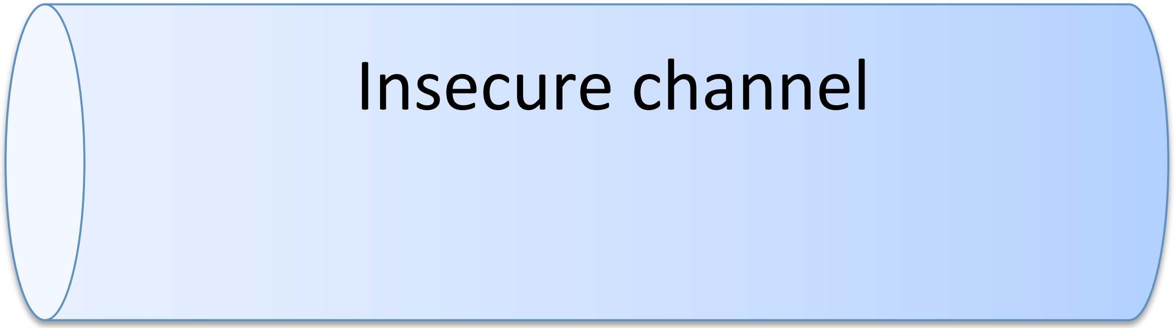
Florian Bergsma   Tibor Jager   Jörg Schwenk

PKC 2015

Horst Görtz Institute for IT-Security

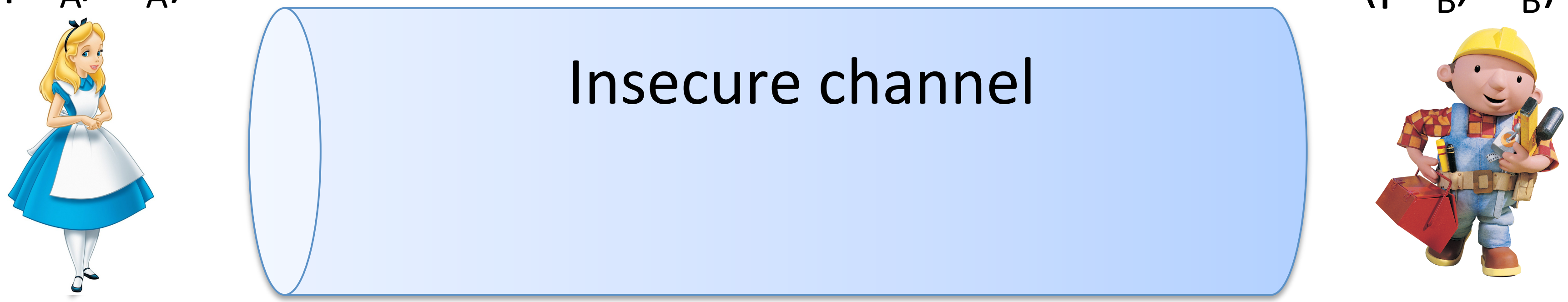
Public-Key Authenticated Key Exchange

Alice



Bob

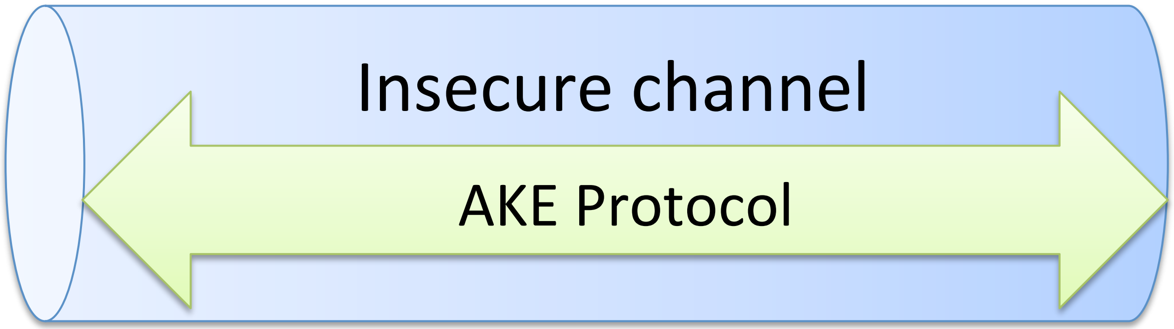
Public-Key Authenticated Key Exchange



Public-Key Authenticated Key Exchange

Alice
(pk_A, sk_A)

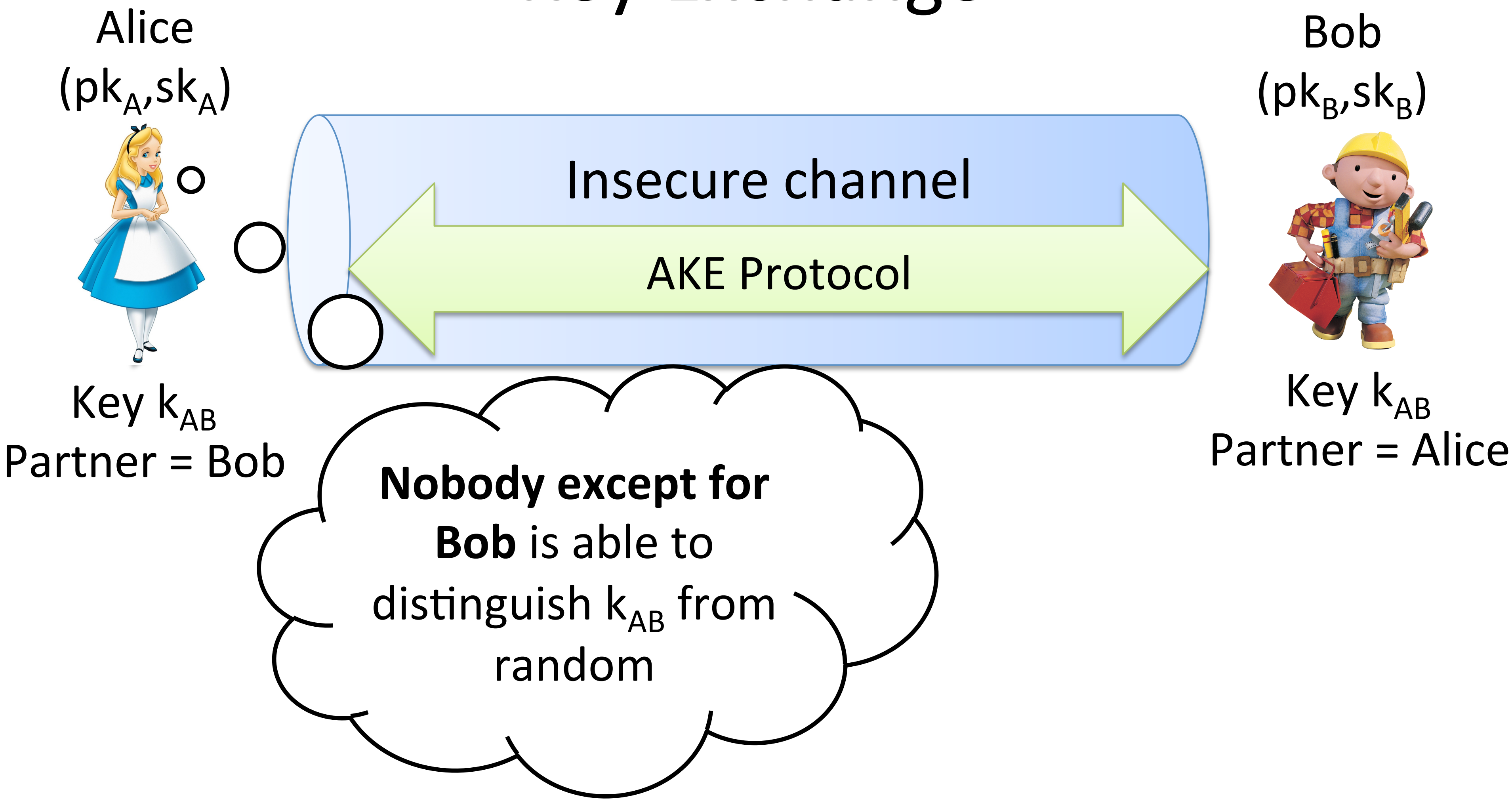
Key k_{AB}



Bob
(pk_B, sk_B)

Key k_{AB}

Public-Key Authenticated Key Exchange



One-Round Key Exchange (ORKE)

(pk_A, sk_A)

(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)

(pk_A, sk_A)

Possibly sent **simultaneously**
(or ***precomputed)***

$m_A = f(pk_B, sk_A, r_A)$ →

← $m_B = f(pk_A, sk_B, r_B)$

(pk_B, sk_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 **simultaneously** (or **precomputed*)***

$m_A = f(pk_B, sk_A, r_A)$ →

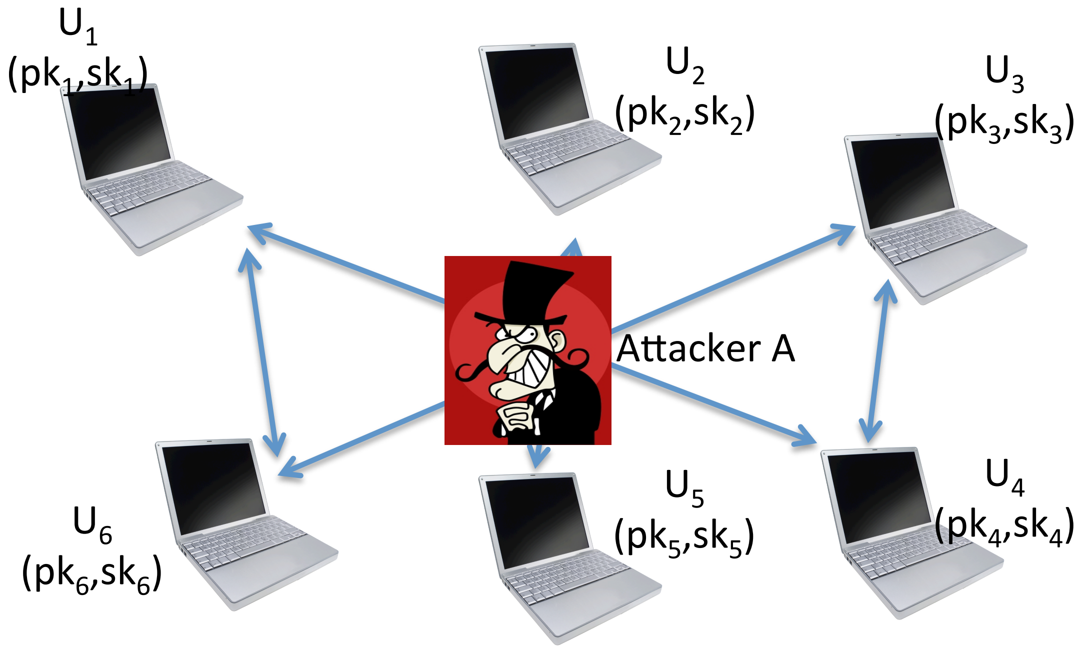
← $m_B = f(pk_A, sk_B, r_B)$

(pk_B, sk_B)

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

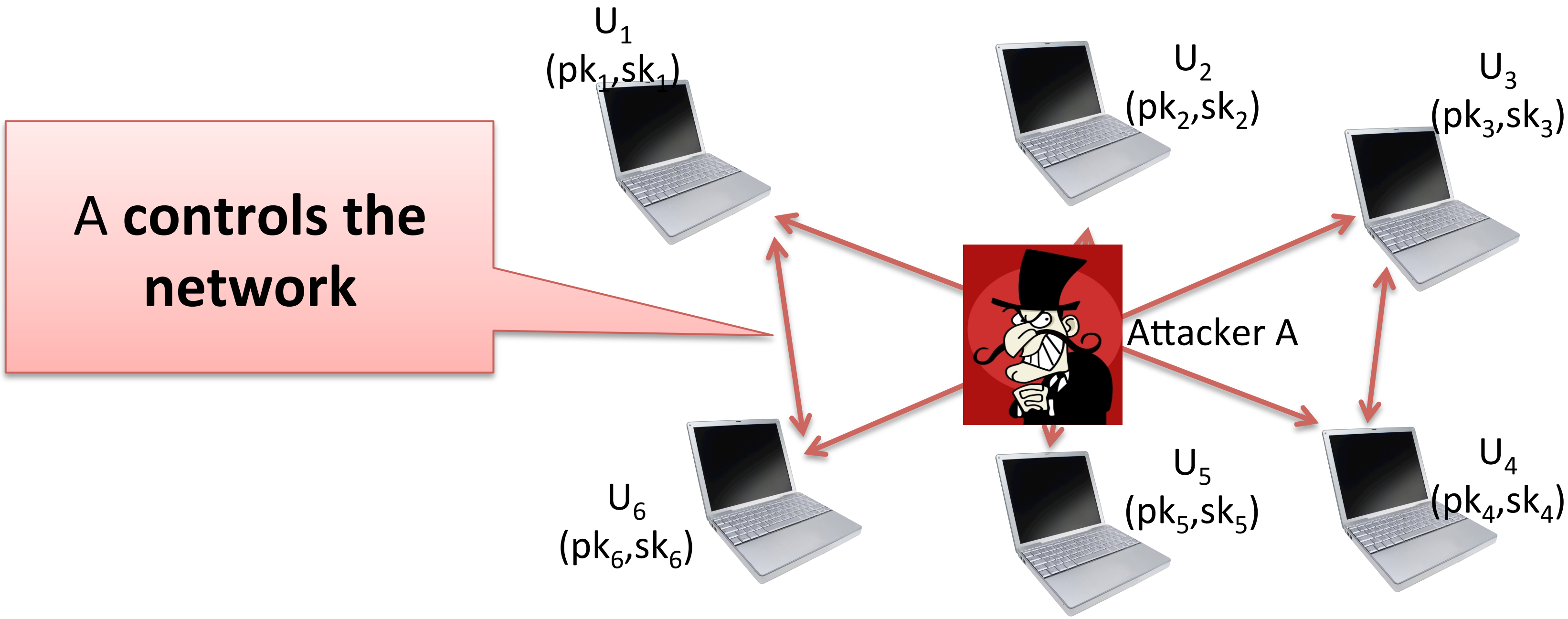
- **Simple** design and implementation
- **Quick** key establishment in at most one RTT

Security Analysis of AKE Protocols



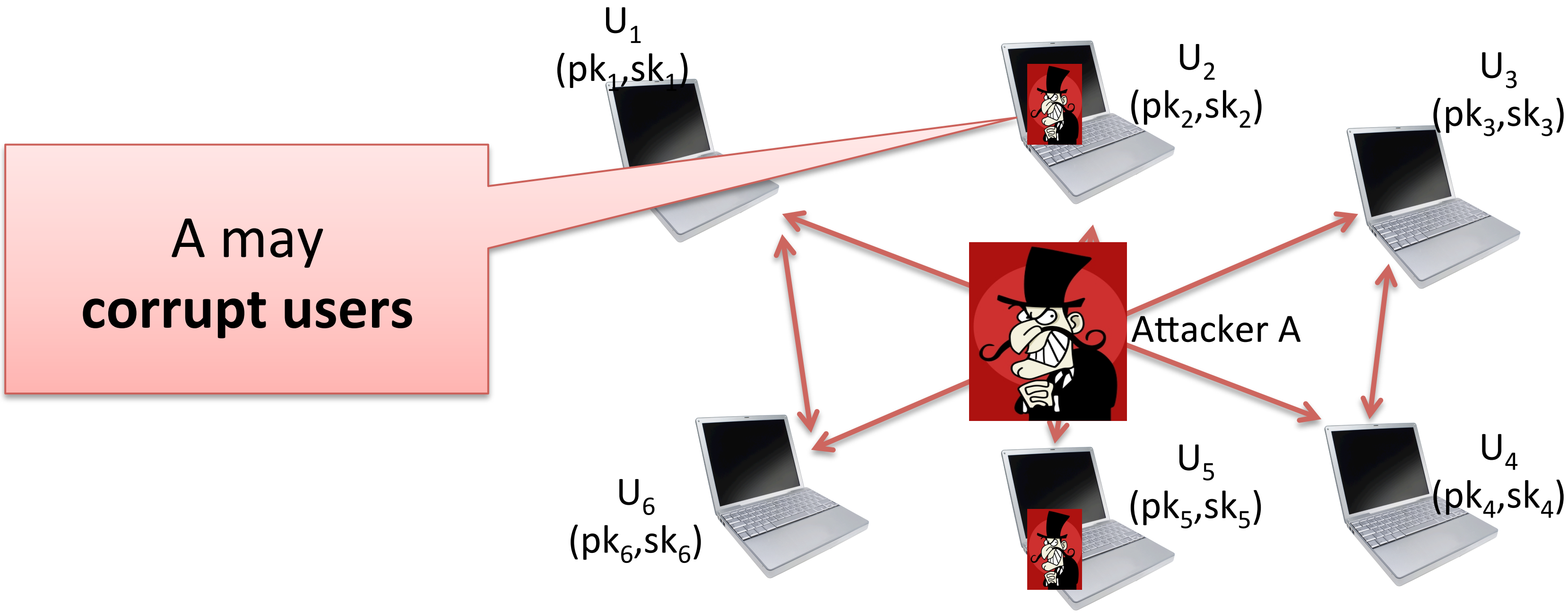
- Provide A with "execution environment" that formalizes A's capabilities

Security Analysis of AKE Protocols



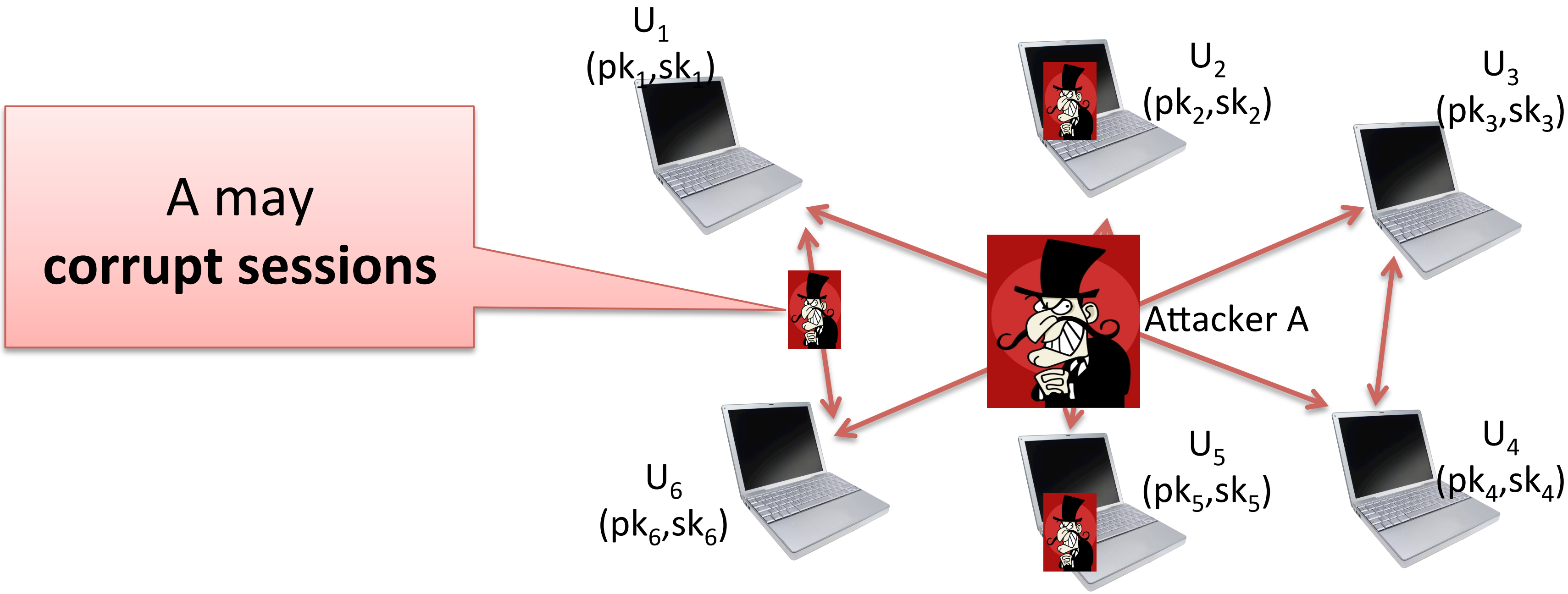
- Provide A with "execution environment" that formalizes A's capabilities

Security Analysis of AKE Protocols



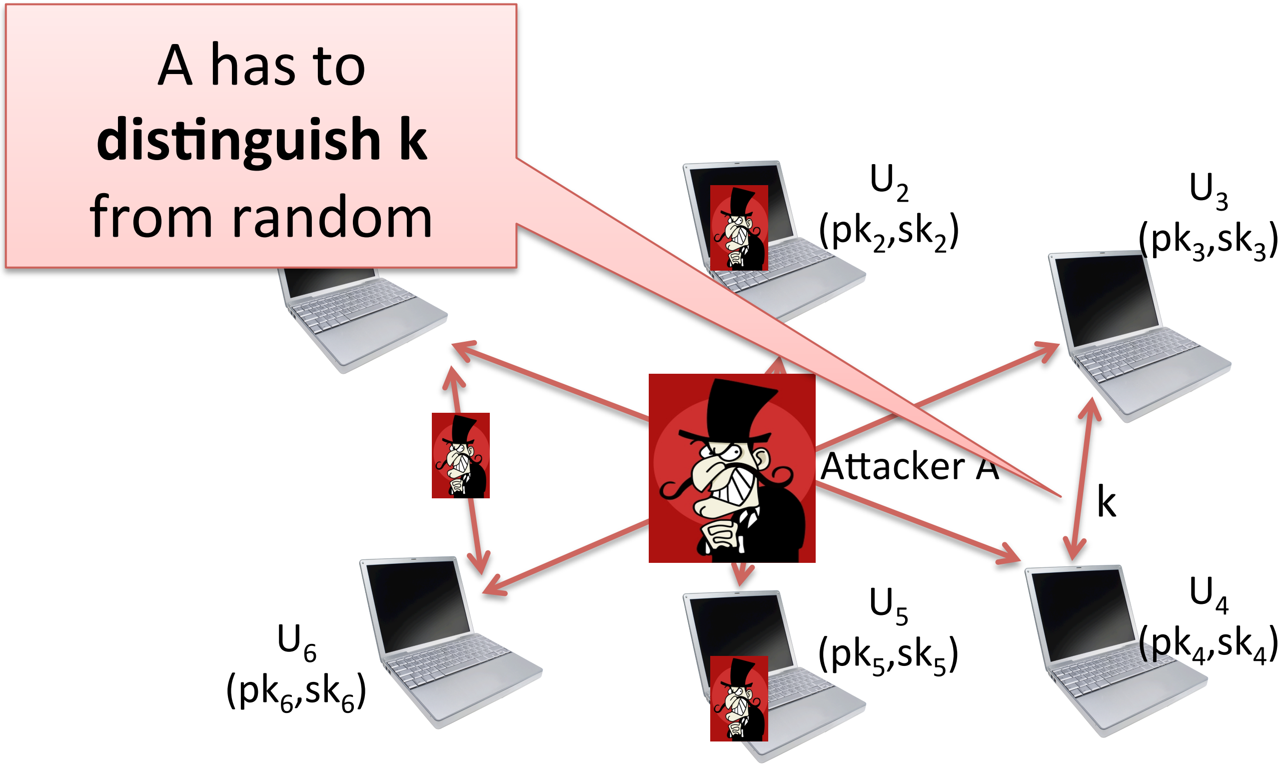
- Provide A with "execution environment" that formalizes A's capabilities

Security Analysis of AKE Protocols



- Provide A with "execution environment" that formalizes A's capabilities

Security Analysis of AKE Protocols

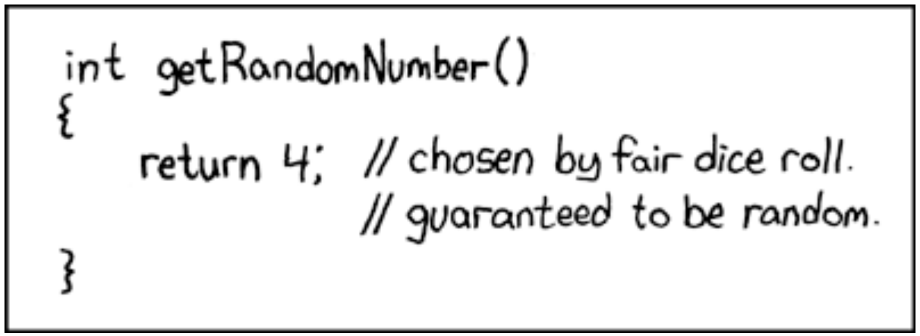


- Provide A with “execution environment” that formalizes A’s capabilities

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)



https://xkcd.com/221/

“eCK Security”

[LLM07]

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

(pk_A, sk_A)

(pk_B, sk_B)

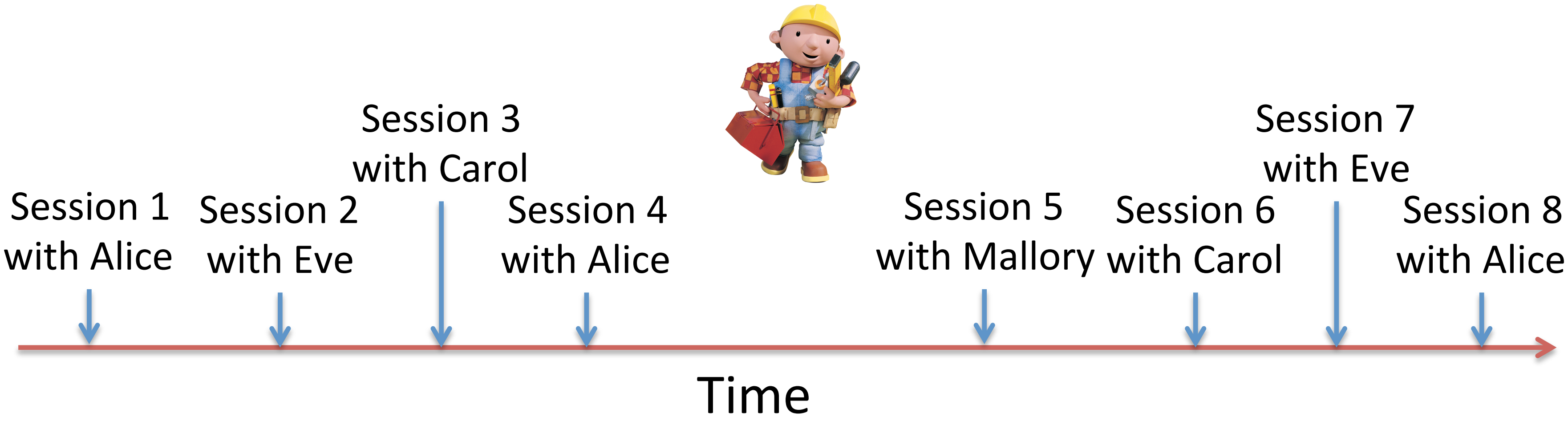
$m_A = f(pk_B, sk_A, \mathbf{r_A})$ →

$m_B = f(pk_A, sk_B, \mathbf{r_B})$ ←

$g(pk_B, sk_A, \mathbf{r_A}, m_B) = k_{AB} = g(pk_A, sk_B, \mathbf{r_B}, m_A)$

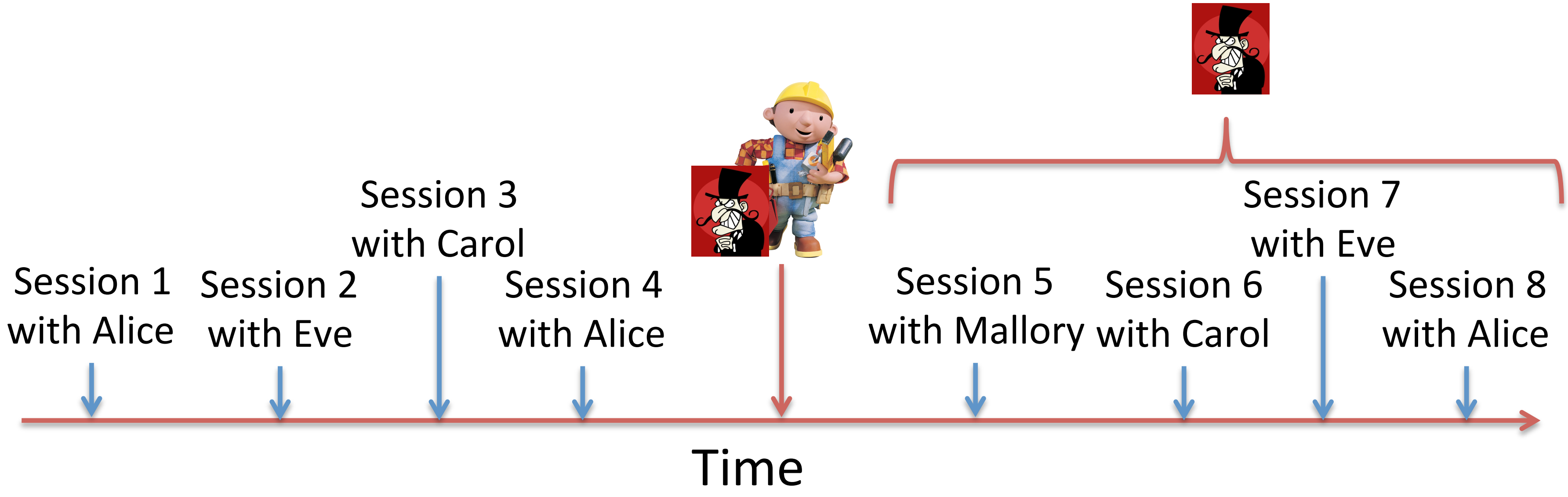
Forward Security (PFS)

(Diffie, van Oorschot, Wiener, DESI 1992)



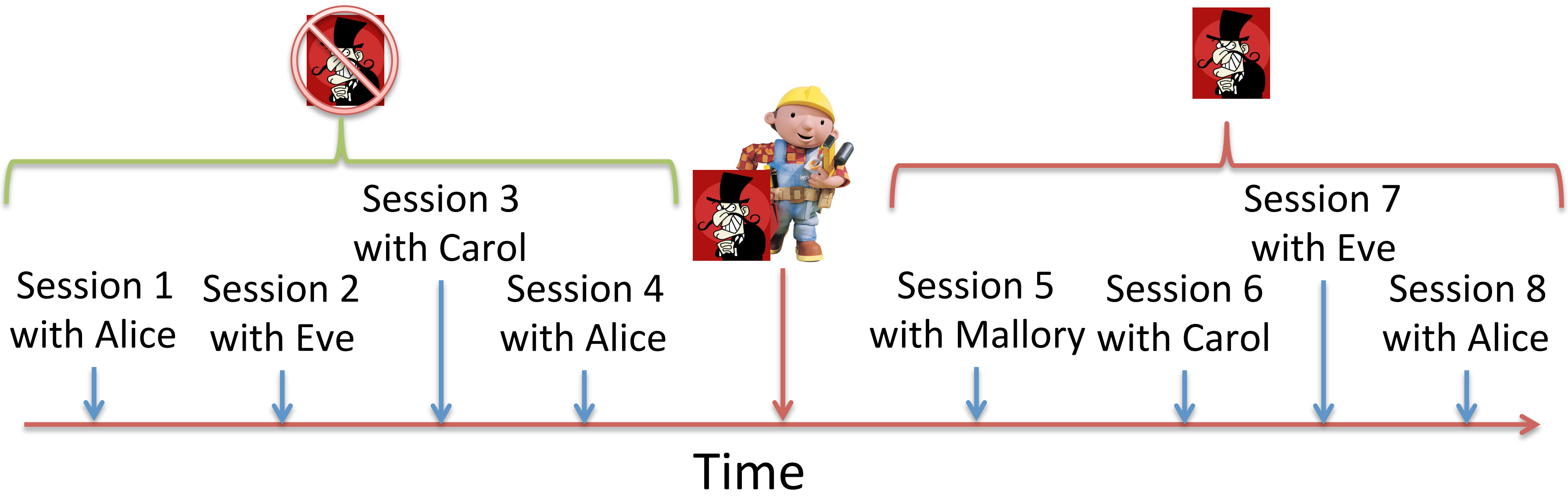
Forward Security (PFS)

(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

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

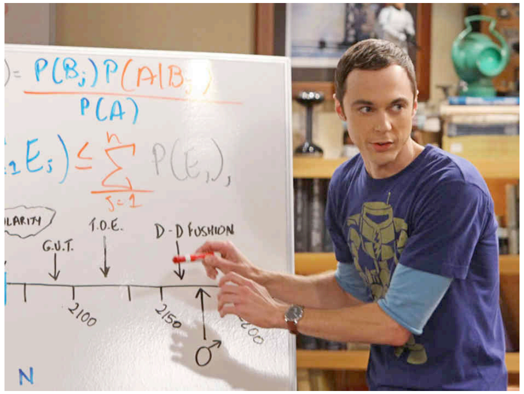
The Difficulty of Forward Security in the eCK Model

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

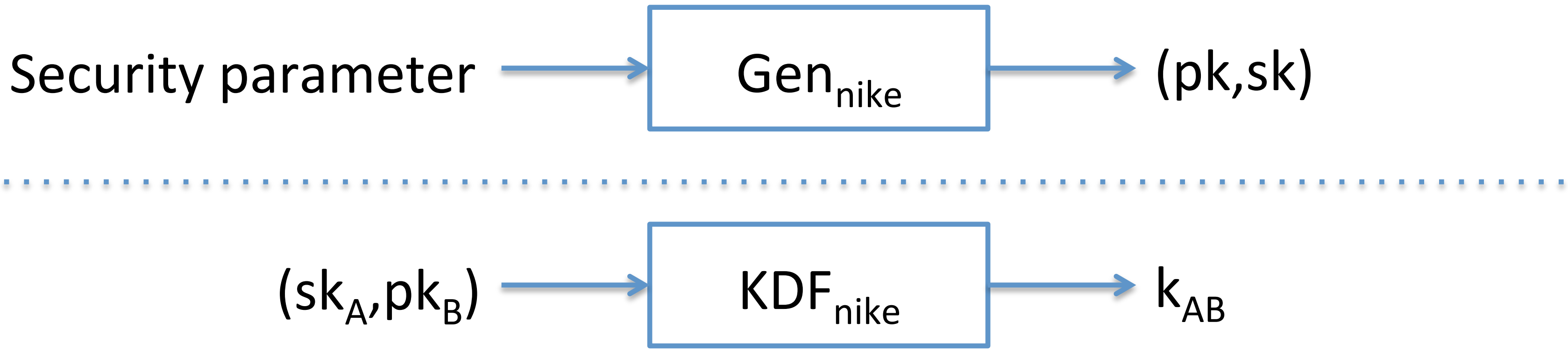
Contributions



- eCK-PFS secure key exchange
  - One-round (ORKE)
  - First from **generic assumptions**
    - Signature scheme
    - Pseudorandom function
    - Non-interactive key exchange
    - **First not based on discrete log** type assumption
  - Without Random Oracles
  - Relatively **efficient**
  - **Simple** construction and proof

Non-Interactive Key Exchange (NIKE)

(Diffie, Hellman `76; Freire, Hofheinz, Kiltz, Paterson, PKC `13)



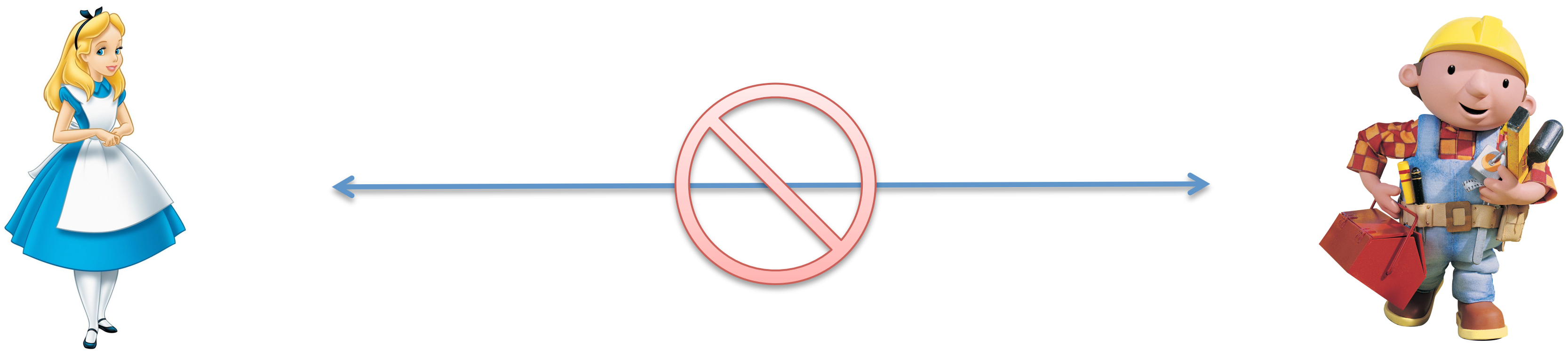
Non-Interactive Key Exchange (NIKE)

(Diffie, Hellman `76; Freire, Hofheinz, Kiltz, Paterson, PKC `13)

Security parameter → Gen_{nike} → (pk,sk)

(sk_A, pk_B) → KDF_{nike} → k_{AB}

(pk_A, sk_A) (pk_B, sk_B)



$\text{KDF}_{nike}(pk_B, sk_A) \;=\; k_{AB} \;=\; \text{KDF}_{nike}(pk_A, sk_B)$

Our Protocol

$pk_A := (pk_{A,sig}, pk_{A,nike})$

$pk_B := (pk_{B,sig}, pk_{B,nike})$

Our Protocol

$pk_A := (pk_{A,sig}, pk_{A,nike})$ $pk_B := (pk_{B,sig}, pk_{B,nike})$

$(pk'_A, sk'_A) \leftarrow NIKEGen(1^k, r_A)$ $(pk'_B, sk'_B) \leftarrow NIKEGen(1^k, r_B)$

$m_A = (pk'_A, sig_A(pk'_A))$ →

$m_B = (pk'_B, sig_B(pk'_B))$ ←

$KDF_{orke}(pk_B, sk_A, m_B, r_A) = k_{AB} = KDF_{orke}(pk_A, sk_B, m_A, r_B)$

Our Protocol

$pk_A := (pk_{A,sig}, pk_{A,nike})$ $pk_B := (pk_{B,sig}, pk_{B,nike})$

$(pk'_A, sk'_A) \leftarrow \text{NIKEGen}(1^k, r_A)$ $(pk'_B, sk'_B) \leftarrow \text{NIKEGen}(1^k, r_B)$

$m_A = (pk'_A,\ sig_A(pk'_A))$ →

← $m_B = (pk'_B,\ sig_B(pk'_B))$

$$KDF_{orke}(pk_B, sk_A, m_B, r_A) \;=\; k_{AB} \;=\; KDF_{orke}(pk_A, sk_B, m_A, r_B)$$

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 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)

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)

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)

Idea of KDF_{orke}

Alice **essentially** computes:

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

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

- 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: $(pk_B, sk_A, (pk_B', sig_B), (pk_A', sig_A))$

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

Output k

Generic Construction

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

  Standard security definitions

- 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

	Standard Model	PFS	weak PFS	KCI	exp. per party	pairing evaluations	Security model
$\mathcal{TS}1$ [21]	✗	✗	✗	✗	1	-	BR^1
$\mathcal{TS}3$ [21]	✓	✓	✓	✗	3	-	BR^1
MQV	✗	✗	✓	✗	1	-	CK
HMQV	✗	✗	✓	✓	2	-	CK
KEA	✗	✗	✓	✓	2	-	CK
P1 [6]	✓	✗	✗	✓	8	2	CK
P2 [6]	✓	✗	✓	✓	10	2	CK
NAXOS	✗	✗	✓	✓	4	-	eCK
Okamoto	✓ $+\pi$PRF	✗	✓	✓	8	-	eCK
NAXOS^2_{pfs}	✗	✓	✓	✓	4	-	$eCK\text{-}PFS$
ORKE^3	✗ (NIKE)	✓	✓	✓	5	-	$eCK\text{-}PFS$
ORKE^4	✓	✓	✓	✓	16	12	$eCK\text{-}PFS$