On the Provable Security of the Dragonfly protocol

Jean Lancrenon[1] **Marjan Škrobot**[1]

[1] Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg

ISC 2015

SnT securityandtrust.lu

uni.lu UNIVERSITÉ DU LUXEMBOURG

Agence Nationale de la Recherche ANR

Fonds National de la Recherche Luxembourg

Outline

1. PAKEs
2. Dragonfly
3. Results
4. Conclusion

Password Authenticated Key Exchange

PAKE Problem:

Password Authenticated Key Exchange

PAKE Problem:

- Setup: Shared low-entropy secret (password)

Password Authenticated Key Exchange

PAKE Problem:

- Setup: Shared low-entropy secret (password)
- Goal: High-entropy session key

Password Authenticated Key Exchange

PAKE Problem:

- Setup: Shared low-entropy secret (password)
- Goal: High-entropy session key
- Without PKI

Password Authenticated Key Exchange

PAKE Problem:

- Setup: Shared low-entropy secret (password)
- Goal: High-entropy session key
- Without PKI
- Only password for authentication

Password Authenticated Key Exchange

PAKE Problem:

- Setup: Shared low-entropy secret (password)
- Goal: High-entropy session key
- Without PKI
- Only password for authentication
- Prevent offline-dictionary attacks

Password Authenticated Key Exchange

PAKE Problem:

- Setup: Shared low-entropy secret (password)
- Goal: High-entropy session key
- Without PKI
- Only password for authentication
- Prevent offline-dictionary attacks
- Limit online-guessing attacks

Design Techniques

Typical approaches for designing *efficient* PAKEs in (ROM):

Design Techniques

Typical approaches for designing *efficient* PAKEs in (ROM):

1. "EKE-style"

$$\xrightarrow{\quad E_{pw}(g^x) \quad}$$

$$\xleftarrow{\quad E_{pw}(g^y) \quad}$$

Design Techniques

Typical approaches for designing *efficient* PAKEs in (ROM):

1. "EKE-style"

$$\xrightarrow{\quad E_{pw}(g^x) \quad}$$

$$\xleftarrow{\quad E_{pw}(g^y) \quad}$$

2. "SPEKE-style"

$$\xrightarrow{\quad (H(pw))^x \quad}$$

$$\xleftarrow{\quad (H(pw))^y \quad}$$

Design Techniques

Typical approaches for designing *efficient* PAKEs in (ROM):

1. "EKE-style"

$$\xrightarrow{\qquad E_{pw}(g^x) \qquad}$$

$$\xleftarrow{\qquad E_{pw}(g^y) \qquad}$$

2. "SPEKE-style"

$$\xrightarrow{\qquad (H(pw))^x \qquad}$$

$$\xleftarrow{\qquad (H(pw))^y \qquad}$$

3. "J-PAKE-style"

$$\xrightarrow{\qquad (D_1)^{xpw},\ \pi_1 \qquad}$$

$$\xleftarrow{\qquad (D_2)^{ypw},\ \pi_2 \qquad}$$

Security Models for PAKE

PAKE Security Models:

Security Models for PAKE

PAKE Security Models:

1. Indistinguishability-Based Model [BR93,95]
   - Find-then-Guess [BPR00]
   - Real-or-Random [AFP05]

Security Models for PAKE

PAKE Security Models:

1. Indistinguishability-Based Model [BR93,95]
   - Find-then-Guess [BPR00]
   - Real-or-Random [AFP05]
2. Simulation-Based Model [S99]
   - Modified Shoup's model [BMP00]
   - Plain model PAKEs [GL01]

Security Models for PAKE

PAKE Security Models:

1. Indistinguishability-Based Model [BR93,95]
   - Find-then-Guess [BPR00]
   - Real-or-Random [AFP05]
2. Simulation-Based Model [S99]
   - Modified Shoup's model [BMP00]
   - Plain model PAKEs [GL01]
3. Universal Composability Model [CK02]
   - UC for PAKE [CHKLM05]

Security Models for PAKE

PAKE Security Models:

1. Indistinguishability-Based Model [BR93,95]
   - **Find-then-Guess [BPR00]**
   - Real-or-Random [AFP05]
2. Simulation-Based Model [S99]
   - Modified Shoup’s model [BMP00]
   - Plain model PAKEs [GL01]
3. Universal Composability Model [CK02]
   - UC for PAKE [CHKLM05]

Find-then-Guess BPR Model

Queries available to PPT adversary $\mathcal{A}$:

- **Send**(U^i, M) - message exchange
- **Execute**(C^i, S^j) - eavesdropping
- **Reveal**(U^i) - leakage of the session key
- **Corrupt**(U) - leakage of the long term secret*
- **Test**(U^i) - semantic security of the session key

Find-then-Guess BPR Model

Queries available to PPT adversary $\mathcal{A}$:

- **Send**(U^i, M) - message exchange
- **Execute**(C^i, S^j) - eavesdropping
- **Reveal**(U^i) - leakage of the session key
- **Corrupt**(U) - leakage of the long term secret*
- **Test**(U^i) - semantic security of the session key

What security means in BPR model?

Find-then-Guess BPR Model

Queries available to PPT adversary $\mathcal{A}$:

- **Send**(U^i, M) - message exchange
- **Execute**(C^i, S^j) - eavesdropping
- **Reveal**(U^i) - leakage of the session key
- **Corrupt**(U) - leakage of the long term secret*
- **Test**(U^i) - semantic security of the session key

What security means in BPR model?

Definition

Protocol P is forward secure PAKE if for all PPT adversaries $\mathcal{A}$ making at most n_{se} online attempts, where N is the size of the dictionary and C is a constant

$$\mathbf{Adv}_{\mathrm{P}}^{ake}(\mathcal{A}) \leq \frac{C \cdot n_{se}}{N} + \varepsilon \ . \tag{1}$$

Motivation

Why Dragonfly?

Motivation

Why Dragonfly?

- Submitted for standard in IETF (patent free)
    - Dragonfly PAKE
    - PSK (PWD) for IKE - RFC 6617 (Experimental), 2012
    - EAP-PWD - RFC 5931 (Informational), 2010
    - TLS-PWD

Motivation

Why Dragonfly?

- Submitted for standard in IETF (patent free)
  - Dragonfly PAKE
  - PSK (PWD) for IKE - RFC 6617 (Experimental), 2012
  - EAP-PWD - RFC 5931 (Informational), 2010
  - TLS-PWD
- Fully symmetric (no strict roles)

Motivation

Why Dragonfly?

- Submitted for standard in IETF (patent free)
  - Dragonfly PAKE
  - PSK (PWD) for IKE - RFC 6617 (Experimental), 2012
  - EAP-PWD - RFC 5931 (Informational), 2010
  - TLS-PWD
- Fully symmetric (no strict roles)
- Follows SPEKE design approach

Motivation

Why Dragonfly?

- Submitted for standard in IETF (patent free)
    - Dragonfly PAKE
    - PSK (PWD) for IKE - RFC 6617 (Experimental), 2012
    - EAP-PWD - RFC 5931 (Informational), 2010
    - TLS-PWD
- Fully symmetric (no strict roles)
- Follows SPEKE design approach
- Without security proof

Motivation

Why Dragonfly?

- Submitted for standard in IETF (patent free)
  - Dragonfly PAKE
  - PSK (PWD) for IKE - RFC 6617 (Experimental), 2012
  - EAP-PWD - RFC 5931 (Informational), 2010
  - TLS-PWD
- Fully symmetric (no strict roles)
- Follows SPEKE design approach
- Without security proof
- Stirred some controversy

Dragonfly draft specifications

Client	**Initialization**	Server

Public: $\mathbb{G}$, p, q; $H_0, H_2 : \{0,1\}^* \rightarrow \{0,1\}^k$; $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{2k}$;

$\pi \in Passwords$; $seed := H_0(C, S, \pi, c)_{max,min}$; $PW := H\&P(seed, l_1)$.

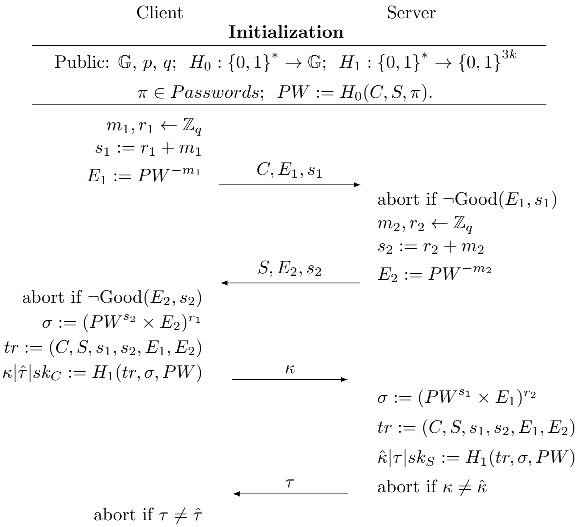
$m_1, r_1 \leftarrow \mathbb{Z}_q$		$m_2, r_2 \leftarrow \mathbb{Z}_q$
$s_1 := r_1 + m_1$		$s_2 := r_2 + m_2$
$E_1 := PW^{-m_1}$		$E_2 := PW^{-m_2}$
	$\xrightarrow{C, E_1, s_1}$	
	$\xleftarrow{S, E_2, s_2}$	
abort if $\neg\mathrm{Good}(E_2, s_2)$		abort if $\neg\mathrm{Good}(E_1, s_1)$
$\sigma := (PW^{s_2} \times E_2)^{r_1}$		$\sigma := (PW^{s_1} \times E_1)^{r_2}$
$kck \vert sk_C := H_1(\sigma, l_2)$		$kck \vert sk_S := H_1(\sigma, l_2)$
$\kappa := H_2(kck, C, s_1, s_2, E_1, E_2)$		$\tau := H_2(kck, S, s_2, s_1, E_2, E_1)$
$\hat{\tau} := H_2(kck, S, s_2, s_1, E_2, E_1)$		$\hat{\kappa} := H_2(kck, C, s_1, s_2, E_1, E_2)$
	$\xrightarrow{\kappa}$	
	$\xleftarrow{\tau}$	
abort if $\tau \neq \hat{\tau}$		abort if $\kappa \neq \hat{\kappa}$

Dragonfly draft specifications

Client		Server
	Initialization	
	Public: $\mathbb{G}$, p, q; $H_0, H_2 : \{0,1\}^* \to \{0,1\}^k$; $H_1 : \{0,1\}^* \to \{0,1\}^{2k}$;	
	$\pi \in Passwords$; $seed := H_0(C, S, \pi, c)_{max,min}$; $PW := H\&P(seed, l_1)$.	
$m_1, r_1 \leftarrow \mathbb{Z}_q$		$m_2, r_2 \leftarrow \mathbb{Z}_q$
$s_1 := r_1 + m_1$		$s_2 := r_2 + m_2$
$E_1 := PW^{-m_1}$		$E_2 := PW^{-m_2}$
	$\xrightarrow{C, E_1, s_1}$	
	$\xleftarrow{S, E_2, s_2}$	
abort if $\neg\text{Good}(E_2, s_2)$		abort if $\neg\text{Good}(E_1, s_1)$
$\sigma := (PW^{s_2} \times E_2)^{r_1}$		$\sigma := (PW^{s_1} \times E_1)^{r_2}$
$kck \mid sk_C := H_1(\sigma, l_2)$		$kck \mid sk_S := H_1(\sigma, l_2)$
$\kappa := H_2(kck, C, s_1, s_2, E_1, E_2)$		$\tau := H_2(kck, S, s_2, s_1, E_2, E_1)$
$\hat{\tau} := H_2(kck, S, s_2, s_1, E_2, E_1)$		$\hat{\kappa} := H_2(kck, C, s_1, s_2, E_1, E_2)$
	$\xrightarrow{\kappa}$	
	$\xleftarrow{\tau}$	
abort if $\tau \neq \hat{\tau}$		abort if $\kappa \neq \hat{\kappa}$

Provable Secure Dragonfly

Our Dragonfly



Client Server

Initialization

Public: $\mathbb{G}$, p, q; $H_0 : \{0,1\}^* \to \mathbb{G}$; $H_1 : \{0,1\}^* \to \{0,1\}^{3k}$

$\pi \in Passwords$; $PW := H_0(C, S, \pi)$.

$m_1, r_1 \leftarrow \mathbb{Z}_q$

$s_1 := r_1 + m_1$

$E_1 := PW^{-m_1}$ $\xrightarrow{C, E_1, s_1}$

abort if $\neg$Good(E_1, s_1)

$m_2, r_2 \leftarrow \mathbb{Z}_q$

$s_2 := r_2 + m_2$

$\xleftarrow{S, E_2, s_2}$ $E_2 := PW^{-m_2}$

abort if $\neg$Good(E_2, s_2)

$\sigma := (PW^{s_2} \times E_2)^{r_1}$

$tr := (C, S, s_1, s_2, E_1, E_2)$

$\kappa | \hat{\tau} | sk_C := H_1(tr, \sigma, PW)$ $\xrightarrow{\kappa}$

$\sigma := (PW^{s_1} \times E_1)^{r_2}$

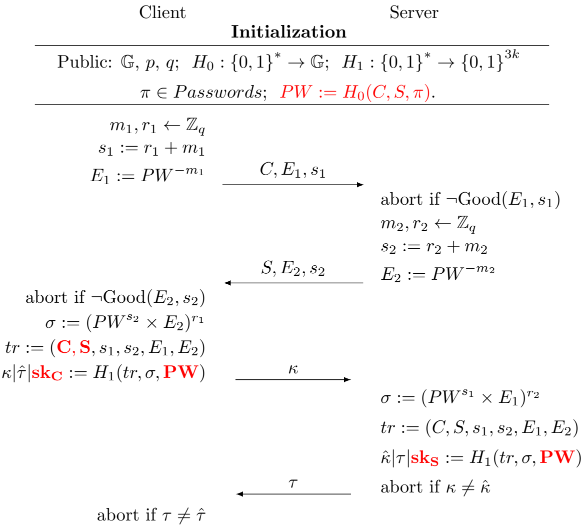
$tr := (C, S, s_1, s_2, E_1, E_2)$

$\hat{\kappa} | \tau | sk_S := H_1(tr, \sigma, PW)$

$\xleftarrow{\tau}$ abort if $\kappa \neq \hat{\kappa}$

abort if $\tau \neq \hat{\tau}$

Our Dragonfly



Client | Server

Initialization

Public: $\mathbb{G}$, p, q; $H_0 : \{0,1\}^* \to \mathbb{G}$; $H_1 : \{0,1\}^* \to \{0,1\}^{3k}$

$\pi \in Passwords$; $PW := H_0(C, S, \pi)$.

Client:

$m_1, r_1 \leftarrow \mathbb{Z}_q$

$s_1 := r_1 + m_1$

$E_1 := PW^{-m_1}$

$\xrightarrow{C, E_1, s_1}$

Server:

abort if $\neg\text{Good}(E_1, s_1)$

$m_2, r_2 \leftarrow \mathbb{Z}_q$

$s_2 := r_2 + m_2$

$E_2 := PW^{-m_2}$

$\xleftarrow{S, E_2, s_2}$

Client:

abort if $\neg\text{Good}(E_2, s_2)$

$\sigma := (PW^{s_2} \times E_2)^{r_1}$

$tr := (\mathbf{C}, \mathbf{S}, s_1, s_2, E_1, E_2)$

$\kappa|\hat{\tau}|\mathbf{sk_C} := H_1(tr, \sigma, \mathbf{PW})$

$\xrightarrow{\kappa}$

Server:

$\sigma := (PW^{s_1} \times E_1)^{r_2}$

$tr := (C, S, s_1, s_2, E_1, E_2)$

$\hat{\kappa}|\tau|\mathbf{sk_S} := H_1(tr, \sigma, \mathbf{PW})$

abort if $\kappa \neq \hat{\kappa}$

$\xleftarrow{\tau}$

Client:

abort if $\tau \neq \hat{\tau}$

Differences between draft and proven variant

Differences:

- "Hunting-and-Pecking" procedure
- Session key computation (sid, PW)
- Confirmation codes (recipient’s identity)
- Symmetric nature:
    - Ordered message exchange
    - Min/Max

The proof of security for Dragonfly

The theorem statement

Theorem

*We consider **Dragonfly** protocol, with a password set of size N. Let $\mathcal{A}$ be an adversary that runs in time at most t, and makes at most n_{se} **Send** queries, n_{ex} **Execute** queries, and n_{h0} and n_{h1} RO queries to H_0 and H_1, resp. Then there exist two algorithms $\mathcal{B}$ and $\mathcal{D}$ running in time t' such that **Adv**$^{ake}_{\texttt{dragonfly}}(\mathcal{A}) \leq T$ where*

$$T := \frac{\mathbf{6n_{se}}}{\mathbf{N}} + \frac{4(n_{se}+n_{ex})(2n_{se}+n_{ex}+n_{h1})}{q^2} + \frac{n_{h0}^2 + 2n_{h1}}{q} + \frac{n_{h1}^2 + 2n_{se}}{2^k} +$$

$$2n_{h1}(1+{n_{se}}^2) \times Succ^{cdh}_{PW,\mathbb{G}}(\mathcal{B}) + 4n_{h0}^3 \times \left(\textbf{\textit{Adv}}^{didh}_{g,\mathbb{G}}(\mathcal{D}) + \frac{n_{h1}^3 + 3n_{se}}{q} \right) \quad (2)$$

and where $t' = O(t + (n_{se}+n_{ex}+n_{ro})t_{exp})$ with t_{exp} being a time required for exponentiation in $\mathbb{G}$.

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries
  - [a]: Randomize session key $H_1'(sid)$ (private oracles)

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries
  - [a]: Randomize session key $H_1'(sid)$ (private oracles)
  - [b]: PW isn't used anymore (except if **Corrupt** query)

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries
    - [a]: Randomize session key $H_1'(sid)$ (private oracles)
    - [b]: PW isn't used anymore (except if **Corrupt** query)
    - [c]: Avoid *lucky* guesses on PW

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries
  - [a]: Randomize session key $H_1'(sid)$ (private oracles)
  - [b]: PW isn't used anymore (except if **Corrupt** query)
  - [c]: Avoid *lucky* guesses on PW ($\mathcal{A}$ has to query H_0)

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries
  - [a]: Randomize session key $H'_1(sid)$ (private oracles)
  - [b]: PW isn't used anymore (except if **Corrupt** query)
  - [c]: Avoid *lucky* guesses on PW ($\mathcal{A}$ has to query H_0)
  - [d]: Avoid *lucky* guesses on authenticators

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries
  - [a]: Randomize session key $H_1'(sid)$ (private oracles)
  - [b]: PW isn't used anymore (except if **Corrupt** query)
  - [c]: Avoid *lucky* guesses on PW ($\mathcal{A}$ has to query H_0)
  - [d]: Avoid *lucky* guesses on authenticators (H_1)

Game hops

- G0: The Dragonfly protocol
- G1: Simulation game
- G2: Force uniqueness and avoid collisions on H_0
- G3: Force random oracle queries
  - [a]: Randomize session key $H_1'(sid)$ (private oracles)
  - [b]: PW isn't used anymore (except if **Corrupt** query)
  - [c]: Avoid *lucky* guesses on PW ($\mathcal{A}$ has to query H_0)
  - [d]: Avoid *lucky* guesses on authenticators (H_1)

AskH1$_3$ event:
$\mathcal{A}$ has to make "correct" combo of H_0 and H_1 queries to win.

The proof of security for Dragonfly

We distinguish four disjoint sub-cases AskH1$_3$:

The proof of security for Dragonfly

We distinguish four disjoint sub-cases $\mathsf{AskH1}_3$:

- **$\mathsf{AskH1\text{-}Passive}_3$** :
  transcript originates from honest execution

The proof of security for Dragonfly

We distinguish four disjoint sub-cases **AskH1**$_3$:

- **AskH1-Passive**$_3$:
  transcript originates from honest execution
- **AskH1-Paired**$_3$:
  $((C, E_1, s_1), (S, E_2, s_2))$ comes from an honest execution,
  while (κ, τ) may come from $\mathcal{A}$;

The proof of security for Dragonfly

We distinguish four disjoint sub-cases **AskH1**$_3$:

- **AskH1-Passive**$_3$:
  transcript originates from honest execution
- **AskH1-Paired**$_3$:
  $((C, E_1, s_1), (S, E_2, s_2))$ comes from an honest execution, while (κ, τ) may come from $\mathcal{A}$;
- **AskH1-withC**$_3$: (S, E_2, s_2) is not from a matching S^j;

The proof of security for Dragonfly

We distinguish four disjoint sub-cases **AskH1**$_3$:

- **AskH1-Passive**$_3$:
  transcript originates from honest execution
- **AskH1-Paired**$_3$:
  $((C, E_1, s_1), (S, E_2, s_2))$ comes from an honest execution, while (κ, τ) may come from $\mathcal{A}$;
- **AskH1-withC**$_3$: (S, E_2, s_2) is not from a matching S^j;
- **AskH1-withS**$_3$: (C, E_1, s_1) is not from a matching C^i.

The proof of security for Dragonfly

We distinguish four disjoint sub-cases **AskH1**$_3$:

- **AskH1-Passive**$_3$:
  transcript originates from honest execution

CDH

- **AskH1-Paired**$_3$:
  $((C, E_1, s_1), (S, E_2, s_2))$ comes from an honest execution, while (κ, τ) may come from $\mathcal{A}$;
- **AskH1-withC**$_3$: (S, E_2, s_2) is not from a matching S^j;
- **AskH1-withS**$_3$: (C, E_1, s_1) is not from a matching C^i.

The proof of security for Dragonfly

We distinguish four disjoint sub-cases AskH1$_3$:

- **AskH1-Passive$_3$** : transcript originates from honest execution

  CDH

- **AskH1-Paired$_3$** : $((C, E_1, s_1), (S, E_2, s_2))$ comes from an honest execution, while (κ, τ) may come from $\mathcal{A}$;

  CDH

- **AskH1-withC$_3$** : (S, E_2, s_2) is not from a matching S^j;
- **AskH1-withS$_3$** : (C, E_1, s_1) is not from a matching C^i.

The proof of security for Dragonfly

We distinguish four disjoint sub-cases **AskH1**$_3$:

- **AskH1-Passive**$_3$:
  transcript originates from honest execution

CDH

- **AskH1-Paired**$_3$:
  $((C, E_1, s_1), (S, E_2, s_2))$ comes from an honest execution, while (κ, τ) may come from $\mathcal{A}$;

CDH

- **AskH1-withC**$_3$: (S, E_2, s_2) is not from a matching S^j;
- **AskH1-withS**$_3$: (C, E_1, s_1) is not from a matching C^i.

DIDH

Security Assumptions

DIDH assumption

Let $IDH_g(X,Y) = g^{1/(x+y)}$.
An algorithm $\mathcal{D}$ is a (t,ε)-DIDH solver if $\mathbf{Adv}^{didh}_{g,\mathbb{G}}(\mathcal{D})$

$$\begin{aligned}\mathbf{Adv}^{didh}_{g,\mathbb{G}}(\mathcal{D}) :=& \\ &\Pr[x,y \leftarrow \mathbb{Z}_q^*, X \leftarrow g^{1/x}; Y \leftarrow g^{1/y}; Z \leftarrow IDH_g(X,Y) : \\ &\qquad\qquad\qquad\qquad \mathcal{D}(X,Y,Z)=1] \\ &\quad - \Pr[x,y,z \in \mathbb{Z}_q^*, X \leftarrow g^{1/x}; Y \leftarrow g^{1/y}; Z \leftarrow g^{1/z} : \\ &\qquad\qquad\qquad\qquad \mathcal{D}(X,Y,Z)=1] \ ,\end{aligned}$$

is bigger than negligible.

The proof of security for Dragonfly

Reduction from DIDH:

The proof of security for Dragonfly

Reduction from DIDH:

- $\mathcal{D}$ chooses 3 distinct random indexes

The proof of security for Dragonfly

Reduction from DIDH:

- $\mathcal{D}$ chooses 3 distinct random indexes
- A triple $\langle X, Y, Z \rangle$ is "plugged" in H_0 outputs

The proof of security for Dragonfly

Reduction from DIDH:

- $\mathcal{D}$ chooses 3 distinct random indexes
- A triple $\langle X, Y, Z \rangle$ is "plugged" in H_0 outputs
- $PW_1 := X^{u_1}$, $PW_2 := Y^{u_2}$, and $PW_3 := Z^{u_3}$

The proof of security for Dragonfly

Reduction from DIDH:

- $\mathcal{D}$ chooses 3 distinct random indexes
- A triple $\langle X, Y, Z \rangle$ is "plugged" in H_0 outputs
- $PW_1 := X^{u_1}$, $PW_2 := Y^{u_2}$, and $PW_3 := Z^{u_3}$
- Extract from H_1 queries: ${E_2}^x$, ${E_2}^y$, and ${E_2}^z$

The proof of security for Dragonfly

Reduction from DIDH:

- $\mathcal{D}$ chooses 3 distinct random indexes
- A triple $\langle X, Y, Z \rangle$ is "plugged" in H_0 outputs
- $PW_1 := X^{u_1}$, $PW_2 := Y^{u_2}$, and $PW_3 := Z^{u_3}$
- Extract from H_1 queries: ${E_2}^x$, ${E_2}^y$, and ${E_2}^z$
- $\mathcal{D}$ wins if ${E_2}^x {E_2}^y = {E_2}^z$

The proof of security for Dragonfly

Reduction from DIDH:

- $\mathcal{D}$ chooses 3 distinct random indexes
- A triple $\langle X, Y, Z \rangle$ is "plugged" in H_0 outputs
- $PW_1 := X^{u_1}$, $PW_2 := Y^{u_2}$, and $PW_3 := Z^{u_3}$
- Extract from H_1 queries: ${E_2}^x$, ${E_2}^y$, and ${E_2}^z$
- $\mathcal{D}$ wins if ${E_2}^x {E_2}^y = {E_2}^z$

$$\Pr[\mathsf{Coll}_C] \leq 2n_{h0}^3 \times \left(\mathsf{Adv}_{g,\mathbb{G}}^{didh}(\mathcal{D}) + \frac{n_{h1}^3 + 3n_{se}}{2q} \right) \quad . \tag{3}$$

The proof of security for Dragonfly

Reduction from DIDH:

- $\mathcal{D}$ chooses 3 distinct random indexes
- A triple $\langle X, Y, Z \rangle$ is "plugged" in H_0 outputs
- $PW_1 := X^{u_1}$, $PW_2 := Y^{u_2}$, and $PW_3 := Z^{u_3}$
- Extract from H_1 queries: ${E_2}^x$, ${E_2}^y$, and ${E_2}^z$
- $\mathcal{D}$ wins if ${E_2}^x {E_2}^y = {E_2}^z$

$$\Pr[\textsf{Coll}_C] \leq 2n_{h0}^3 \times \left(\textsf{Adv}^{didh}_{g,\mathbb{G}}(\mathcal{D}) + \frac{n_{h1}^3 + 3n_{se}}{2q} \right) \quad . \tag{3}$$

$$\Pr[\textsf{AskH1-withC}_4] \leq \frac{\mathbf{2}n_{se}}{N} \quad . \tag{4}$$

Summary of results

Summary of results

- Forward secure in BRP model with ROM

Summary of results

- Forward secure in BRP model with ROM
- Up to 2 password guesses per online attempt

Summary of results

- Forward secure in BRP model with ROM
- Up to 2 password guesses per online attempt
- As secure as SPEKE protocol

Summary of results

- Forward secure in BRP model with ROM
- Up to 2 password guesses per online attempt
- As secure as SPEKE protocol
- Slightly less efficient (4 exp vs. 4 exp + 2 mexp)

Summary of results

- Forward secure in BRP model with ROM
- Up to 2 password guesses per online attempt
- As secure as SPEKE protocol
- Slightly less efficient (4 exp vs. 4 exp + 2 mexp)
- Recommendations: sid in sk and ID in authenticators.