

A Fast and Simple Partially Oblivious PRF, with Applications

Nirvan Tyagi

Sofía Celi

Tom Ristenpart

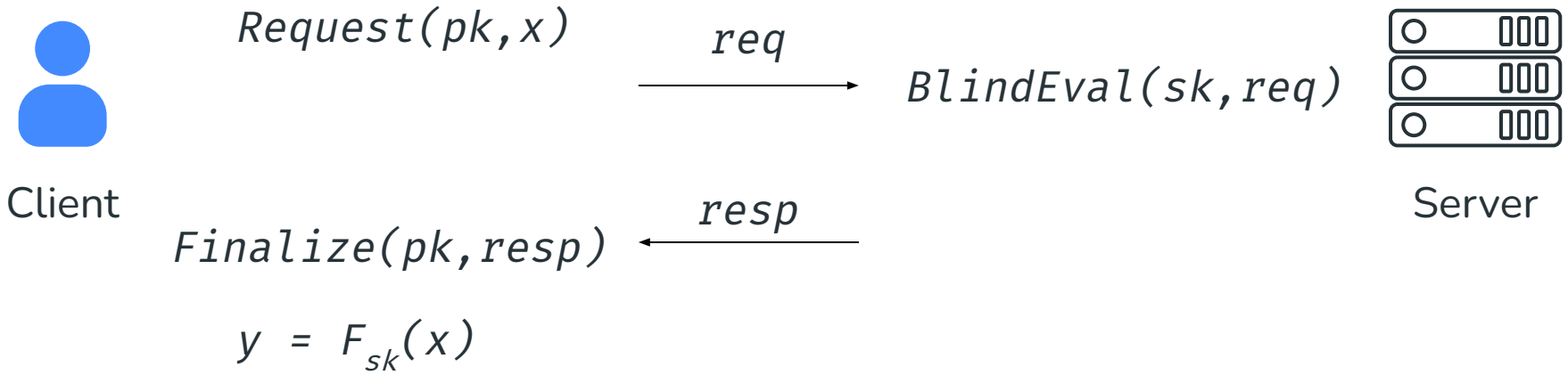
Nick Sullivan

Stefano Tessaro

Chris Wood

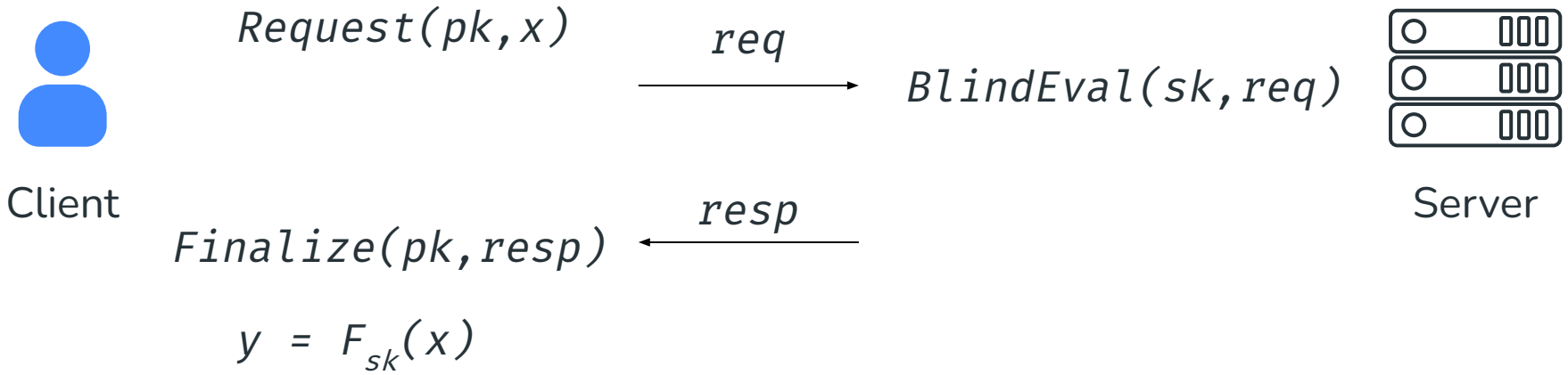
Background: Oblivious PRFs (OPRFs)

An OPRF allows for oblivious evaluation of a pseudorandom function (PRF)



Background: Oblivious PRFs (OPRFs)

- An OPRF allows for oblivious evaluation of a pseudorandom function (PRF)
- Client learns the PRF evaluation and learns nothing else about the PRF key
 - Server holding the PRF key learns nothing about the client input
 - (Optional) Client holds PRF public key to verify evaluation is correct



Background: Oblivious PRFs (OPRFs)

An OPRF allows for oblivious evaluation of a pseudorandom function (PRF)

- Client learns the PRF evaluation and learns nothing else about the PRF key
- Server holding the PRF key learns nothing about the client input
- (Optional) Client holds PRF public key to verify evaluation is correct



Client

Used in many privacy-preserving applications:

- One-time-use authentication tokens [DGSTV PETS '18]
- Bucketized PSI for password breach alerting [LPASCR CCS '19]
- Password-authenticated key exchange [JKX EUROCRYPT '18]
- Online advertisement attribution [SHD ePrint '21]

$y = f(x, sk)$

Background: Standards-track 2HashDH OPRF

[JKK ASIACRYPT '14]

$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$

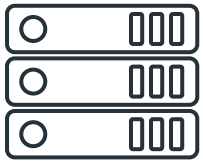


Request(pk=g^{sk}, x)

Client



BlindEval(sk, req)



Server



Finalize(pk, resp)

Background: Standards-track 2HashDH OPRF

[JKK ASIACRYPT '14]

$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$



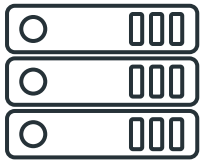
Request(pk=g^{sk}, x)

$r \leftarrow_{\$} Z_p$

Client

$req = H_1(x)^r$

BlindEval(sk, req)



Server

Finalize(pk, resp)



Background: Standards-track 2HashDH OPRF

[JKK ASIACRYPT '14]

$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$



Request(pk=g^{sk}, x)

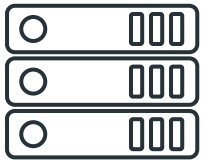
$$r \leftarrow_{\$} Z_p$$

Client

$$req = H_1(x)^r$$

BlindEval(sk, req)

$$B \leftarrow req^{sk}$$



Server

$$resp = B$$

Finalize(pk, resp)

Background: Standards-track 2HashDH OPRF

[JKK ASIACRYPT '14]

$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$



Request(pk=g^{sk}, x)

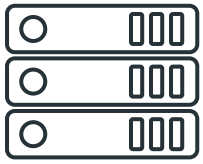
$$r \leftarrow_{\$} Z_p$$

Client

$$req = H_1(x)^r$$

BlindEval(sk, req)

$$B \leftarrow req^{sk}$$



Server

$$resp = B$$

Finalize(pk, resp)

$$B' \leftarrow B^{1/r}$$

$$y = F_{sk}(x) \leftarrow H_2(x, B')$$

Background: Standards-track 2HashDH OPRF

[JKK ASIACRYPT '14]

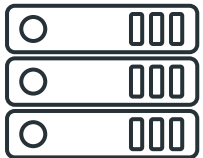
$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$



Request(pk=g^{sk}, x)

$$r \leftarrow_{\$} Z_p$$

$$req = H_1(x)^r$$



BlindEval(sk, req)

Server

$$B \leftarrow req^{sk}$$

$$\pi \leftarrow DLEQ\{pk=g^{sk} \wedge B=req^{sk}\}$$

$$resp = (B, \pi)$$

Finalize(pk, resp)

$$Verify \pi: \exists \alpha, DLEQ\{pk=g^\alpha \wedge B=req^\alpha\}$$

$$B' \leftarrow B^{1/r}$$

$$y = F_{sk}(x) \leftarrow H_2(x, B')$$

Client

Background: Partially Oblivious PRFs (POPRFs)

[ECSJR USENIX Security '15]

A POPRF allows for oblivious evaluation of a pseudorandom function (PRF) on a hybrid input of secret data and public data (called a tag).

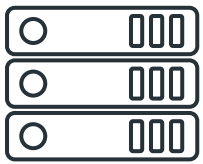


Client

$Request(pk, x)$



$BlindEval(sk, t, req)$



Server

$Finalize(pk, t, resp)$



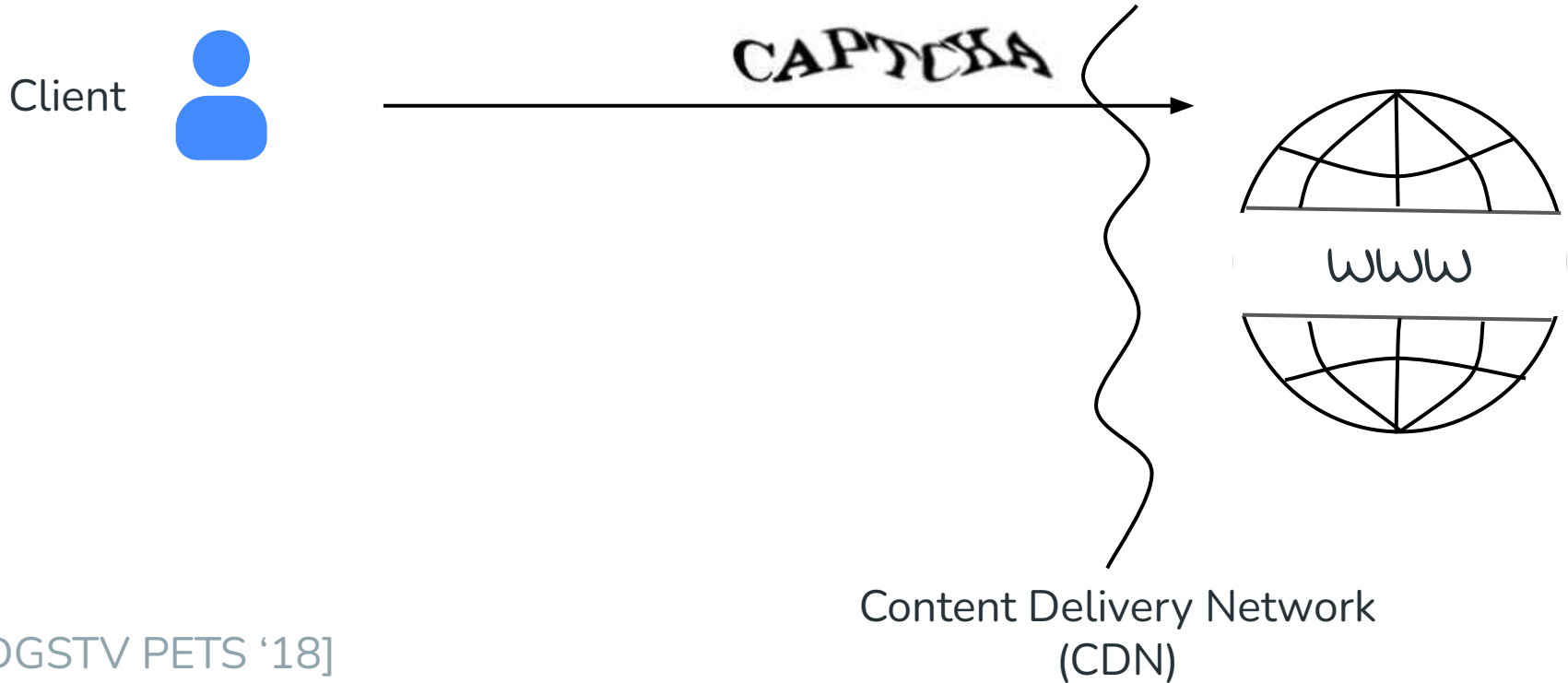
$$y = F_{sk}(x, t)$$

Outline

- Motivation
 - POPRFs simplify use of OPRFs in existing applications
 - Previous candidate POPRFs rely on pairings
- 3HashSDHI: A new, simple POPRF
 - Shares similar structure to standards-track 2HashDH OPRF
- New proof techniques for non-trivial security analysis
 - First proof of closely related partially blind signature scheme

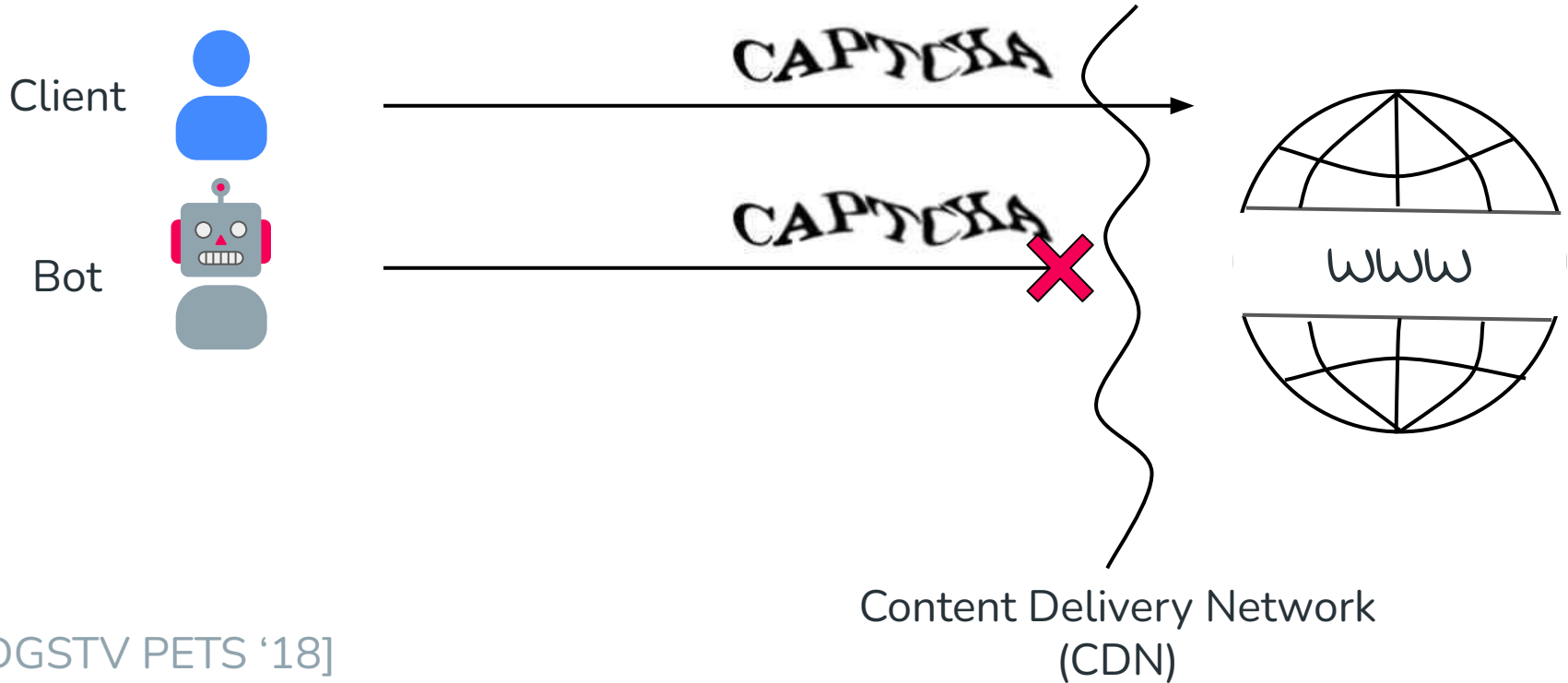
Application: PrivacyPass for bypassing CAPTCHAs

- CAPTCHAs are often used by CDNs to prevent botnets from attacking sites



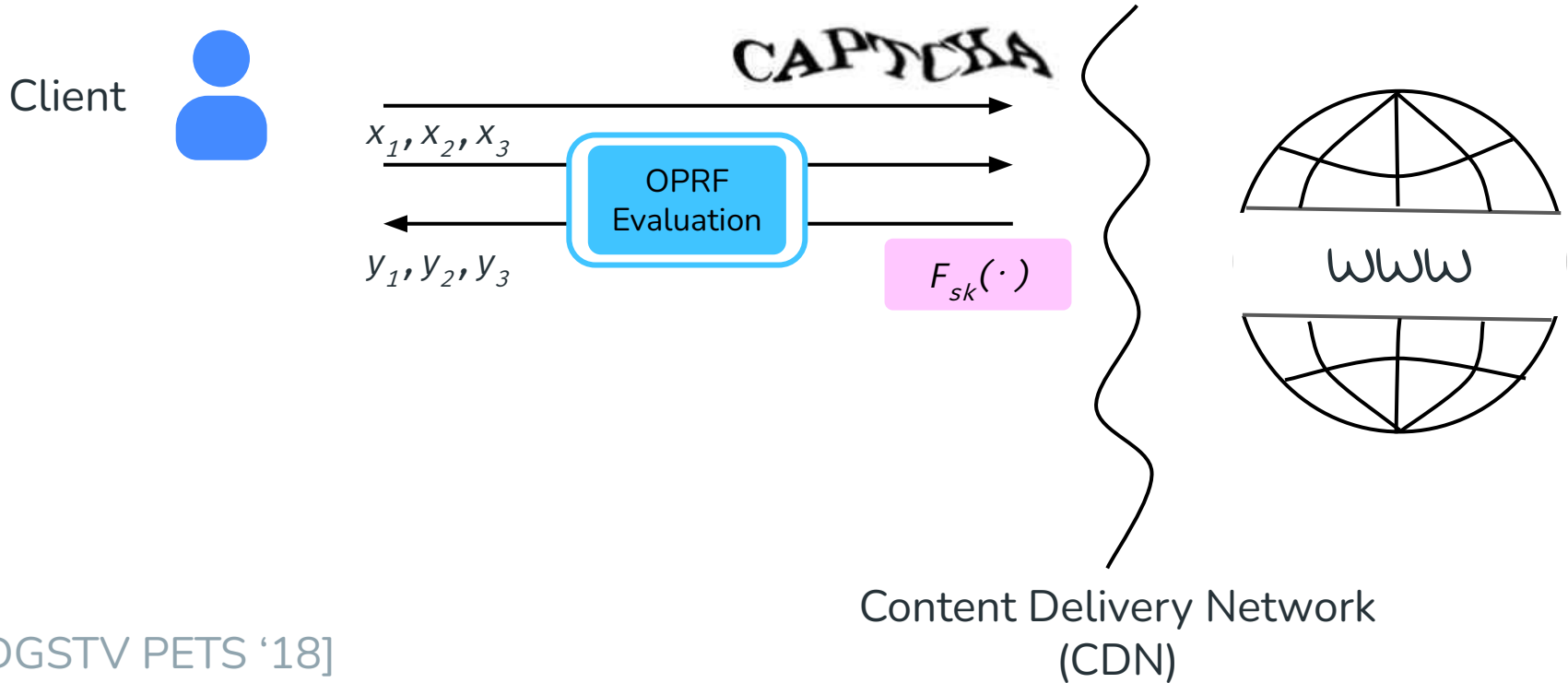
Application: PrivacyPass for bypassing CAPTCHAs

- CAPTCHAs are often used by CDNs to prevent botnets from attacking sites



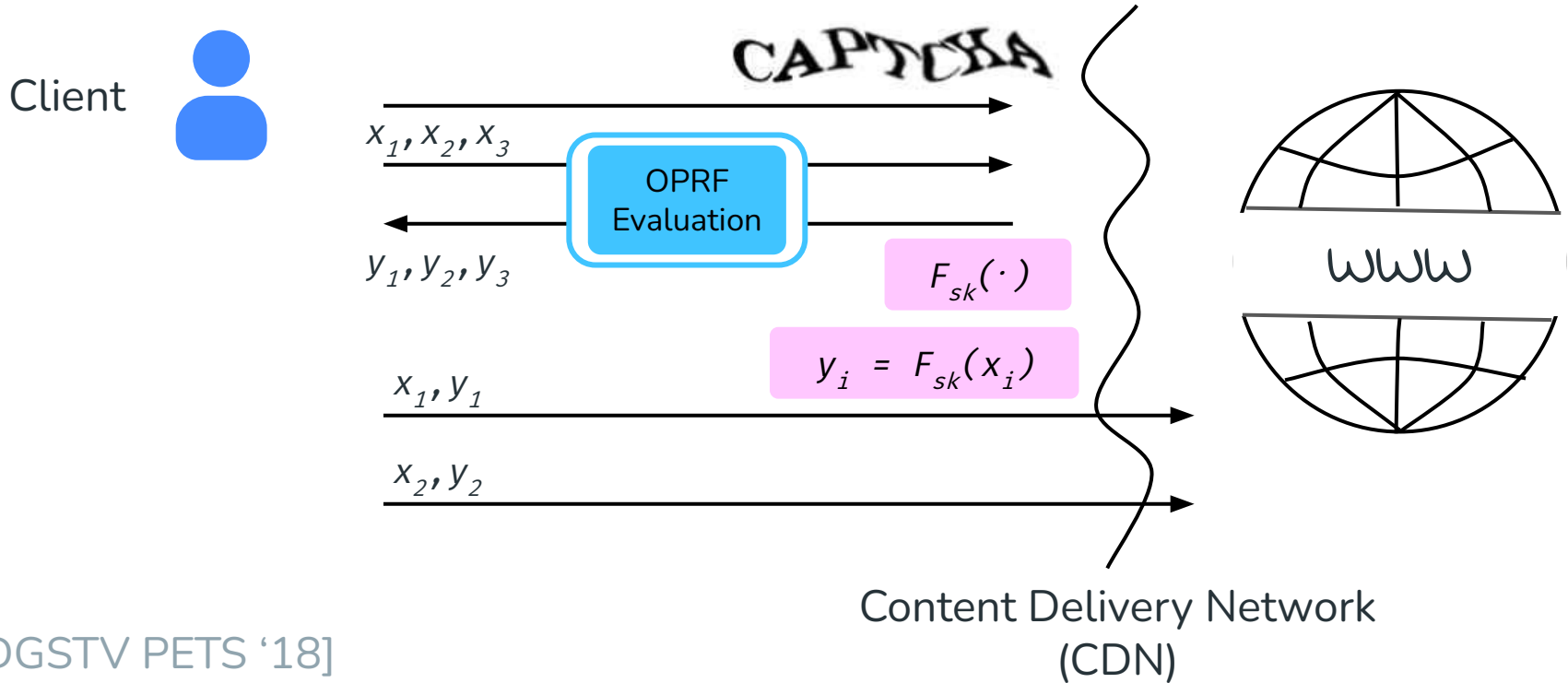
Application: PrivacyPass for bypassing CAPTCHAs

- CAPTCHAs are often used by CDNs to prevent botnets from attacking sites
- After 1 CAPTCHA, client computes OPRF evaluations to use for future auth



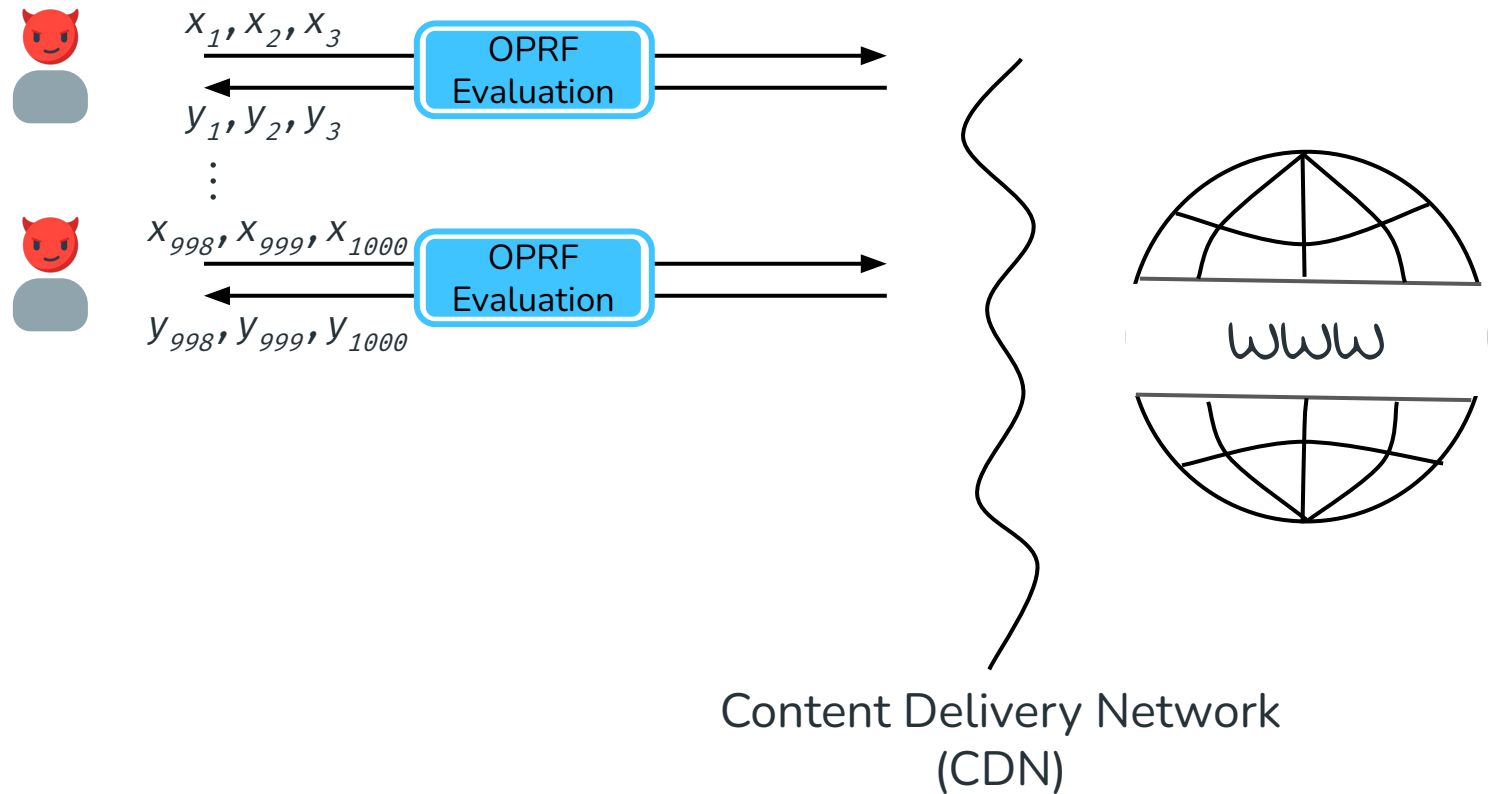
Application: PrivacyPass for bypassing CAPTCHAs

- CAPTCHAs are often used by CDNs to prevent botnets from attacking sites
- After 1 CAPTCHA, client computes OPRF evaluations to use for future auth



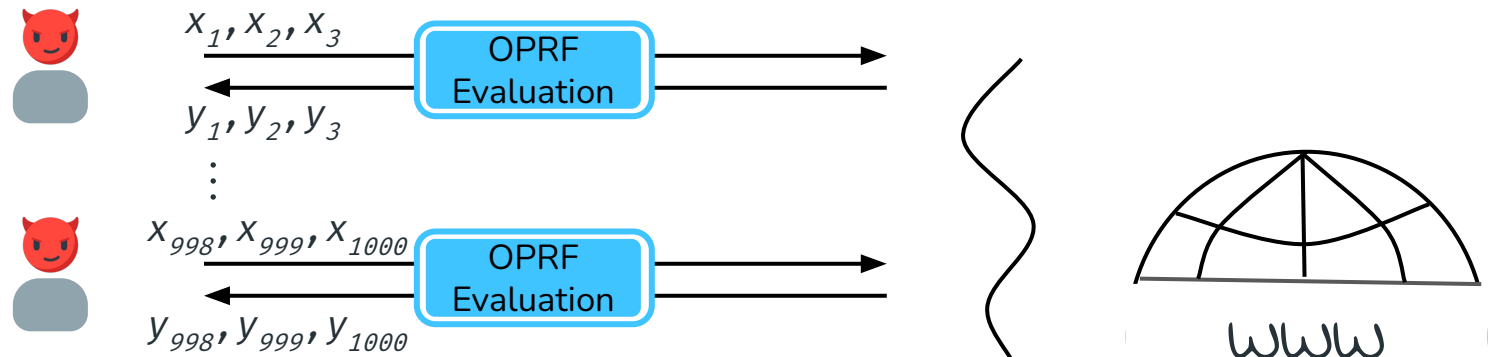
Attack against PrivacyPass: Token hoarding

- Human users collect enough evaluations over time to attack a site

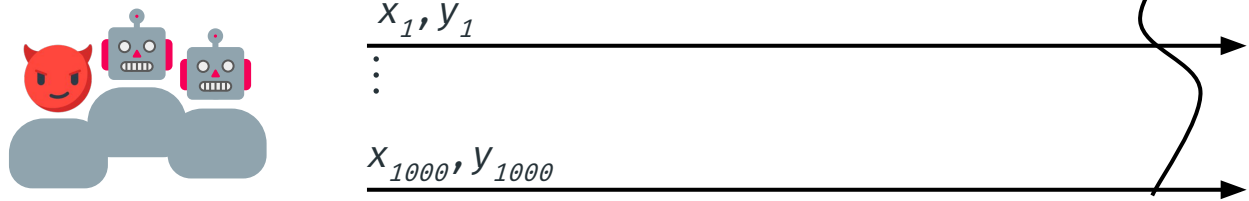


Attack against PrivacyPass: Token hoarding

- Human users collect enough evaluations over time to attack a site



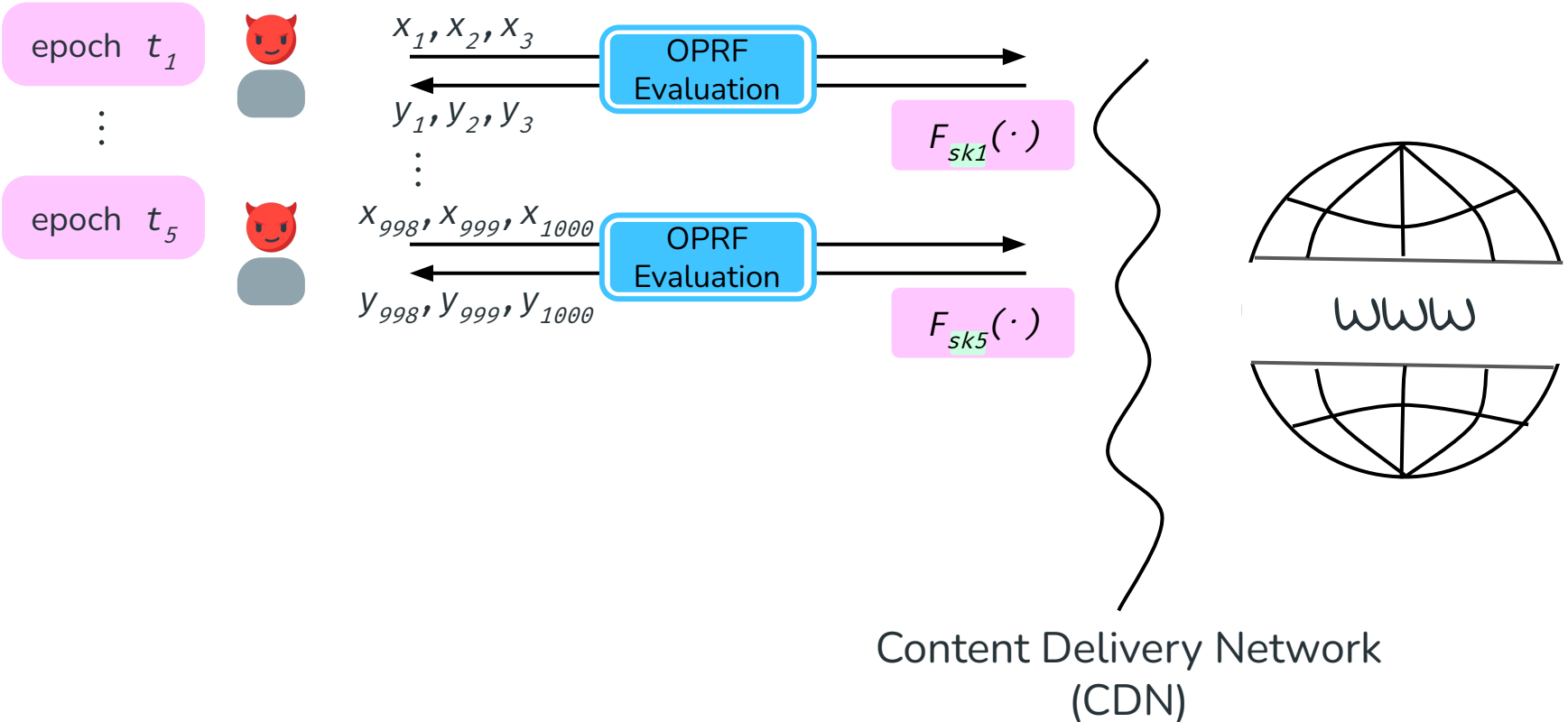
Denial-of-service attack



Content Delivery Network (CDN)

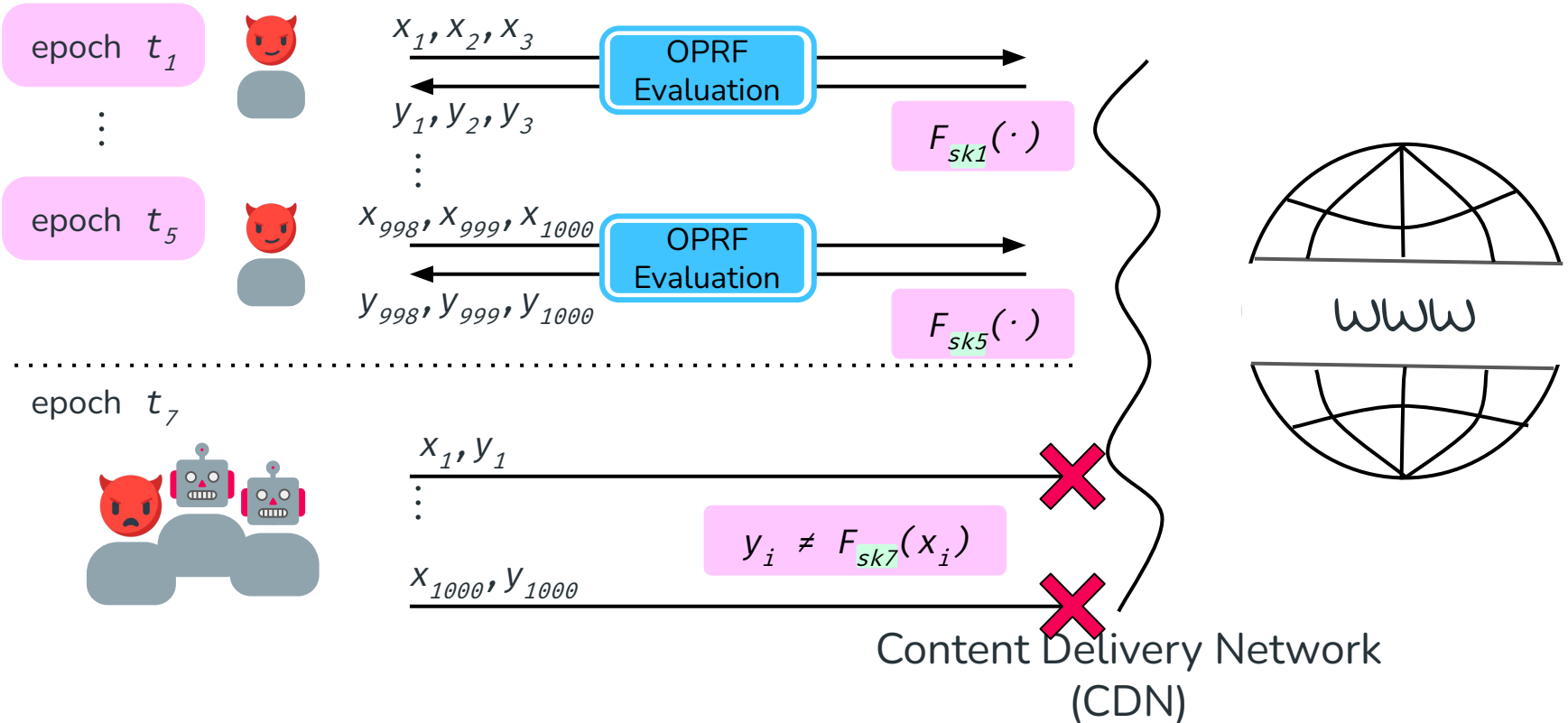
Token hoarding mitigation: OPRF per-epoch key

- Evaluations are provided using separate OPRF keys specified for a time epoch



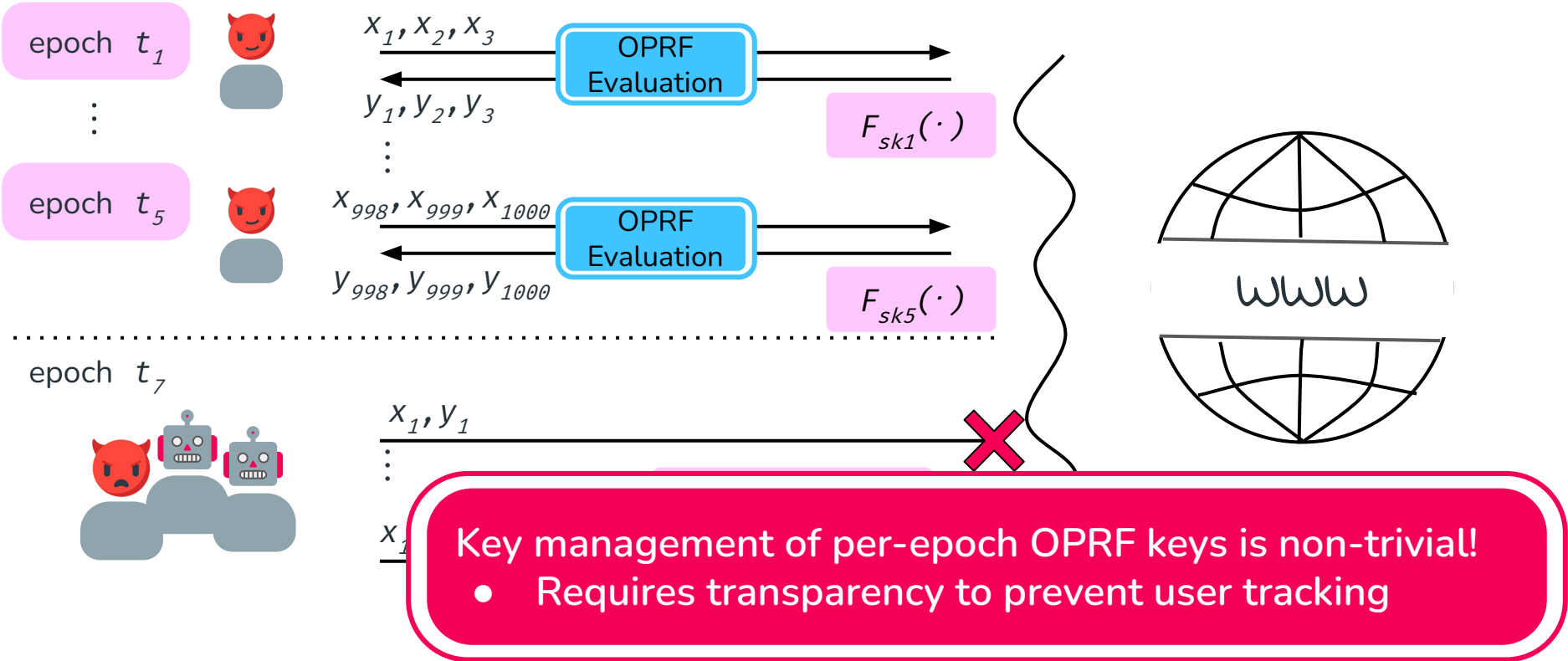
Token hoarding mitigation: OPRF per-epoch key

- Evaluations are provided using separate OPRF keys specified for a time epoch



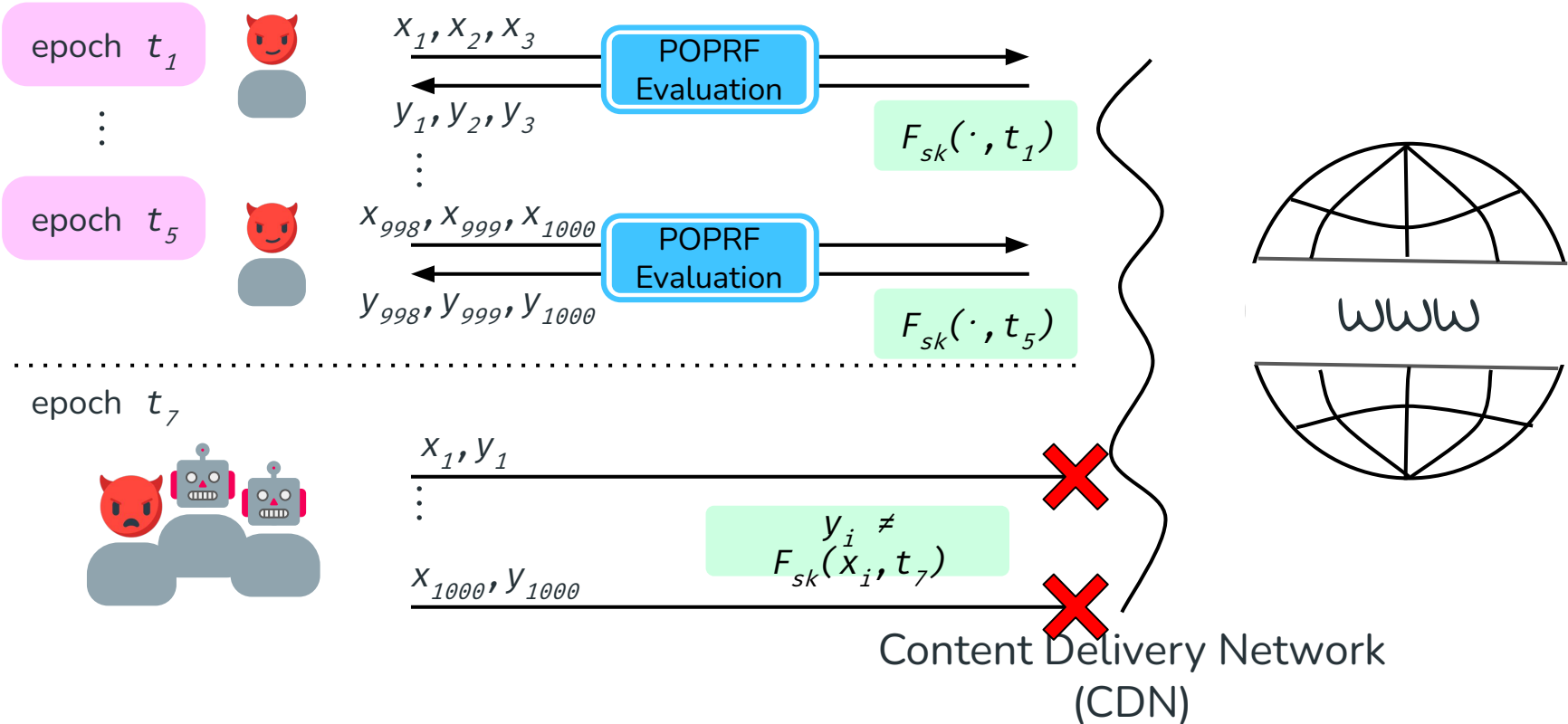
Token hoarding mitigation: OPRF per-epoch key

- Evaluations are provided using separate OPRF keys specified for a time epoch



Token hoarding mitigation: POPRF w/ epoch tag

- Evaluations use a POPRF key and specify the epoch in the public tag



Motivation: POPRFs simplify many OPRF apps

Application	OPRF	POPRF
One-time-use authentication tokens [DGSTV PETS '18]	per-epoch key	epoch tag
Bucketized private set membership for password breach alerting [LPASCR CCS '19]	per-bucket key	bucket tag
Password-authenticated key exchange [JKX EUROCRYPT '18]	per-user key	user tag
Online advertisement attribution [SHD ePrint '21]	per-publisher key	publisher tag

Problem: Existing POPRFs not suitable

Problem: Existing POPRFs not suitable

Pythia POPRF [ECSJR USENIX Security '15]

$$F_{sk}(x, t) = e(H_1(x), H_2(t))^{sk}$$

- First proposal of POPRF
- Relies on pairings
 - Lack of wide library support for pairings (in particular for browsers)

Problem: Existing POPRFs not suitable

Pythia POPRF [ECSJR USENIX Security '15]

$$F_{sk}(x, t) = e(H_1(x), H_2(t))^{sk}$$

- First proposal of POPRF
- Relies on pairings
 - Lack of wide library support for pairings (in particular for browsers)

2HashDH-ext POPRF [JKR ePrint '18]

$$F_{sk}(x, t) = H_2(x, H_1(x)^{H_3(sk, t)})$$

- Generic transform OPRF to POPRF
- Lacks efficient verifiability

Problem: Existing POPRFs not suitable

Pythia POPRF [ECSJR USENIX Security '15]

$$F_{sk}(x, t) = e(H_1(x), H_2(t))^{sk}$$

- First proposal of POPRF
- Relies on pairings
 - Lack of wide library support for pairings (in particular for browsers)

2HashDH-ext POPRF [JKR ePrint '18]

$$F_{sk}(x, t) = H_2(x, H_1(x)^{H_3(sk, t)})$$

- Generic transform OPRF to POPRF
- Lacks efficient verifiability

Can we build a verifiable POPRF from just a discrete-log hard group?

Outline

- Motivation
 - POPRFs simplify use of OPRFs in existing applications
 - Previous candidate POPRFs rely on pairings
- 3HashSDHI: A new, simple POPRF
 - Shares similar structure to standards-track 2HashDH OPRF
- New proof techniques for non-trivial security analysis
 - First proof of closely related partially blind signature scheme

Our construction

3HashSDHI POPRF [This work]

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$

Our construction

3HashSDHI POPRF [This work]

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$

Combines aspects of

2HashDH OPRF [JKK ASIACRYPT '14]

$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$

&

DY PRF [DY PKC '05]

$$F_{sk}(x) = g^{1/(sk+x)}$$

Our construction

3HashSDHI POPRF [This work]

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$

Combines aspects of

2HashDH OPRF [JKK ASIACRYPT '14]

$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$

&

DY PRF [DY PKC '05]

$$F_{sk}(x) = g^{1/(sk+x)}$$

Our construction

3HashSDHI POPRF [This work]

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$

Combines aspects of

2HashDH OPRF [JKK ASIACRYPT '14]

$$F_{sk}(x) = H_2(x, H_1(x)^{sk})$$

&

DY PRF [DY PKC '05]

$$F_{sk}(x) = g^{1/(sk+x)}$$

3HashSDHI: Oblivious evaluation

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$



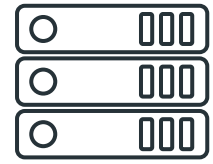
Request(pk=g^{sk}, x)

$r \leftarrow_{\$} Z_p$

Client

$req = H_1(x)^r$

BlindEval(sk, t, req)



Server

Finalize(pk, t, resp)

3HashSDHI: Oblivious evaluation

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$



Request(pk=g^{sk}, x)

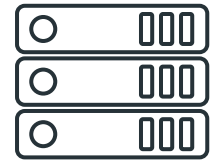
$r \leftarrow_{\$} Z_p$

Client

$req = H_1(x)^r$

BlindEval(sk, t, req)

$B \leftarrow req^{1/(sk+H_3(t))}$



Server

$resp = B$

Finalize(pk, t, resp)

3HashSDHI: Oblivious evaluation

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$



Request(pk=g^{sk}, x)
 $r \leftarrow_{\$} Z_p$

$req = H_1(x)^r$



BlindEval(sk, t, req)
 $B \leftarrow req^{1/(sk+H_3(t))}$

Server

$resp = B$

Finalize(pk, t, resp)

$B' \leftarrow B^{1/r}$
 $y = F_{sk}(x, t) \leftarrow H_2(x, t, B')$

3HashSDHI: Oblivious evaluation

$$F_{sk}(x, t) = H_2(x, t, H_1(x)^{1/(sk+H_3(t))})$$



Request($pk=g^{sk}, x$)
 $r \leftarrow_{\$} Z_p$

$$req = H_1(x)^r$$



BlindEval(sk, t, req)

Server

$$B \leftarrow req^{1/(sk+H_3(t))}$$

$$\pi \leftarrow DLEQ\{pk \cdot g^{H_3(t)} = g^{sk+H_3(t)}\}$$

$$resp = (B, \pi)$$

$$\wedge req = B^{sk+H_3(t)}$$

Finalize($pk, t, resp$)

$$Verify \pi: \exists \alpha, DLEQ\{pk \cdot g^{H_3(t)} = g^\alpha \wedge req = B^\alpha\}$$

$$B' \leftarrow B^{1/r}$$

$$y = F_{sk}(x, t) \leftarrow H_2(x, t, B')$$

3HashSDHI: Oblivious

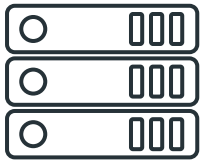
Minimal delta from 2HashDH oblivious evaluation!



Request(pk=g^{sk}, x)

$r \leftarrow_{\$} Z_p$

$req = H_1(x)^r$



BlindEval(sk, t, req)

Server

$B \leftarrow req^{1/(sk+H_3(t))}$

$\pi \leftarrow DLEQ\{pk \cdot g^{H_3(t)} = g^{sk+H_3(t)}\}$

$resp = (B, \pi)$

$\wedge req = B^{sk+H_3(t)}$

Finalize(pk, t, resp)

Verify $\pi: \exists \alpha, DLEQ\{pk \cdot g^{H_3(t)} = g^\alpha \wedge req = B^\alpha\}$

$B' \leftarrow B^{1/r}$

$y = F_{sk}(x, t) \leftarrow H_2(x, t, B')$

Outline

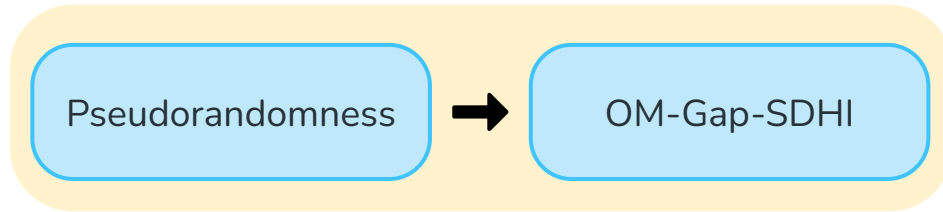
- Motivation
 - POPRFs simplify use of OPRFs in existing applications
 - Previous candidate POPRFs rely on pairings
- 3HashSDHI: A new, simple POPRF
 - Shares similar structure to standards-track 2HashDH OPRF
- New proof techniques for non-trivial security analysis
 - First proof of closely related partially blind signature scheme

Overview: Security of 3HashSDHI

- New property-based security definitions for (P)OPRFs
- Pseudorandomness of 3HashSDHI secure under new assumption
 - One-more Gap Strong Diffie-Hellman Inversion

Overview: Security of 3HashSDHI

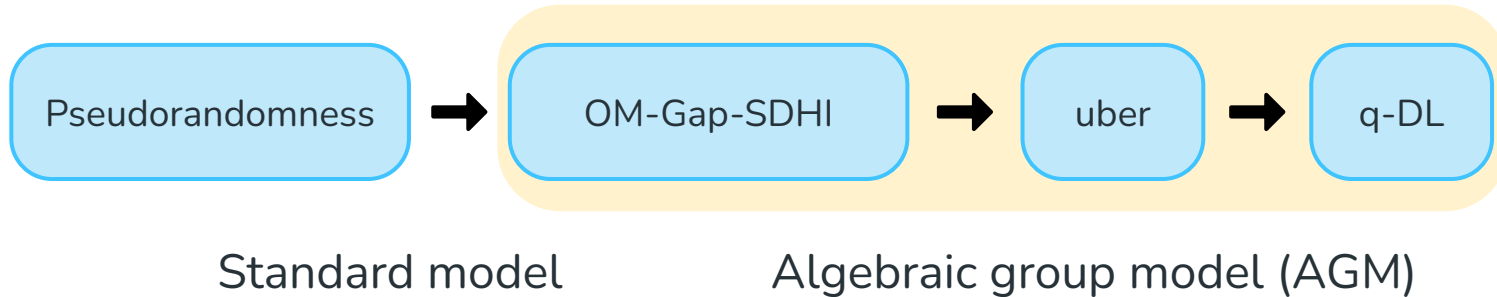
- New property-based security definitions for (P)OPRFs
- Pseudorandomness of 3HashSDHI secure under new assumption
 - One-more Gap Strong Diffie-Hellman Inversion



Standard model

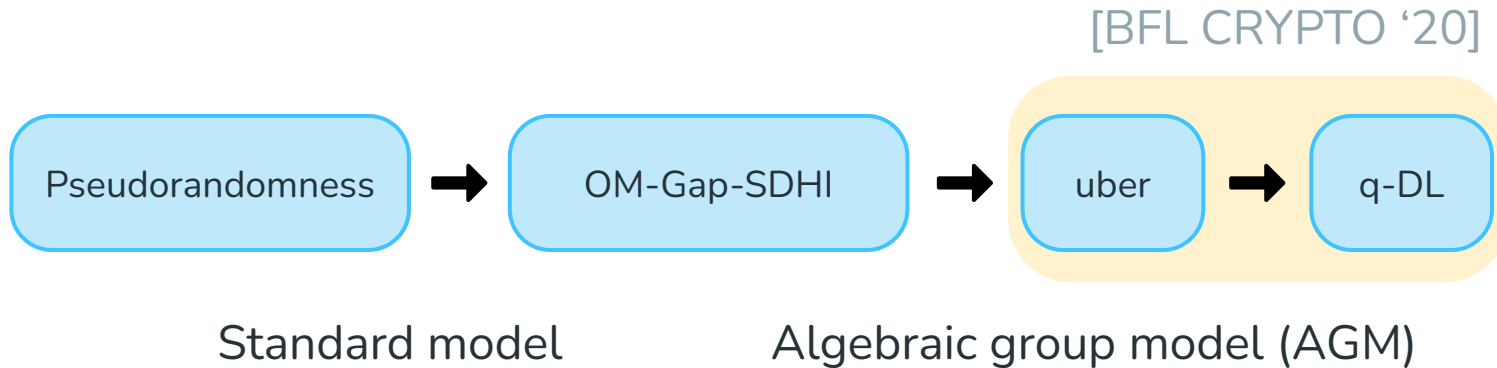
Overview: Security of 3HashSDHI

- New property-based security definitions for (P)OPRFs
- Pseudorandomness of 3HashSDHI secure under new assumption
 - One-more Gap Strong Diffie-Hellman Inversion



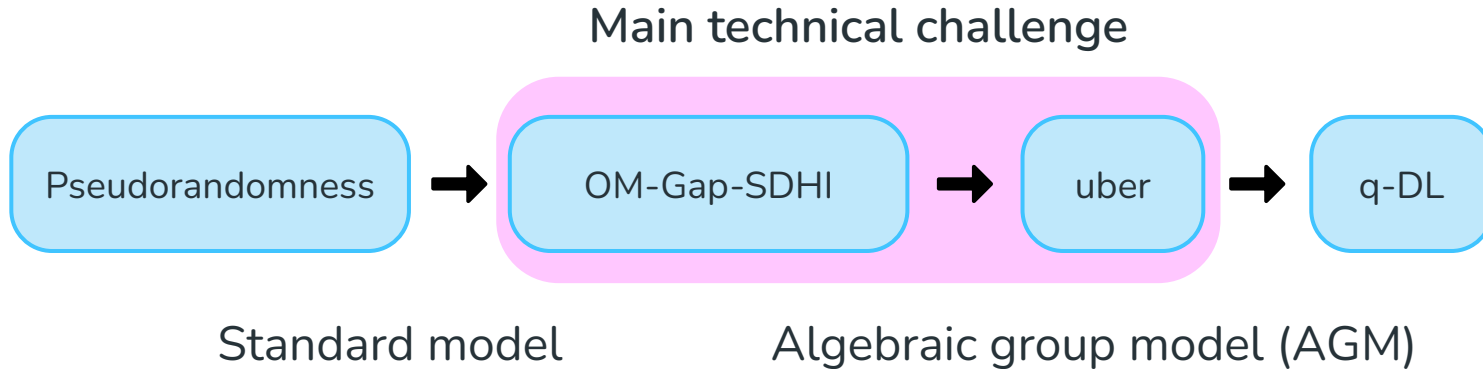
Overview: Security of 3HashSDHI

- New property-based security definitions for (P)OPRFs
- Pseudorandomness of 3HashSDHI secure under new assumption
 - One-more Gap Strong Diffie-Hellman Inversion



Overview: Security of 3HashSDHI

- New property-based security definitions for (P)OPRFs
- Pseudorandomness of 3HashSDHI secure under new assumption
 - One-more Gap Strong Diffie-Hellman Inversion



One-more (Gap) Strong Diffie-Hellman Inversion

OM-SDHI

$$x \leftarrow_{\$} Z_p; \forall t, q_t \leftarrow 0$$

$$y_1, \dots, y_m \leftarrow_{\$} (Z_p)^m$$

$$C_1, \dots, C_l, t \leftarrow_{\$} A^{SDH}(g^x, g^{y_1}, \dots, g^{y_m})$$

A wins if $q_t < l \wedge \forall i, \exists j, C_i = g^{y_j / (x+t)}$

Oracle SDH(B, t)

$$q_t \leftarrow q_t + 1$$

Return $B^{1/(x+t)}$

One-more (Gap) Strong Diffie-Hellman Inversion

OM-SDHI

$x \leftarrow_{\$} Z_p; \forall t, q_t \leftarrow 0$

$y_1, \dots, y_m \leftarrow_{\$} (Z_p)^m$

$C_1, \dots, C_l, t \leftarrow_{\$} A^{SDH}(g^x, g^{y_1}, \dots, g^{y_m})$

A wins if $q_t < l \wedge \forall i, \exists j, C_i = y_j g^{1/(x+t)}$

Oracle $SDH(B, t)$

$q_t \leftarrow q_t + 1$

Return $B^{1/(x+t)}$

Per-tag query counter!

One-more (Gap) Strong Diffie-Hellman Inversion

OM-SDHI

$$x \leftarrow_{\$} Z_p; \forall t, q_t \leftarrow \emptyset$$

$$y_1, \dots, y_m \leftarrow_{\$} (Z_p)^m$$

$$C_1, \dots, C_l, t \leftarrow_{\$} A^{SDH}(g^x, g^{y_1}, \dots, g^{y_m})$$

A wins if $q_t < l \wedge \forall i, \exists j, C_i = g^{y_j / (x+t)}$

- Relation to POPRF security: Per-tag query counter ensures each blind evaluation results in the client learning at most one PRF evaluation restricted to the queried tag

Oracle SDH(B, t)

$q_t \leftarrow q_t + 1$
Return $B^{1/(x+t)}$

Per-tag query counter!

One-more (Gap) Strong Diffie-Hellman Inversion

OM-SDHI

$$x \leftarrow_{\$} Z_p; \forall t, q_t \leftarrow \theta$$

$$y_1, \dots, y_m \leftarrow_{\$} (Z_p)^m$$

$$C_1, \dots, C_l, t \leftarrow_{\$} A^{SDH}(g^x, g^{y_1}, \dots, g^{y_m})$$

A wins if $q_t < l \wedge \forall i, \exists j, C_i = g^{y_j / (x+t)}$

- Relation to POPRF security: Per-tag query counter ensures each blind evaluation results in the client learning at most one PRF evaluation restricted to the queried tag
- Reduction to uber assumption: Argue linear independence of elements given to adversary and winning elements

Oracle SDH(B, t)

$$q_t \leftarrow q_t + 1$$
$$\text{Return } B^{1/(x+t)}$$

Per-tag query counter!

One-more (Gap) Strong Diffie-Hellman Inversion

OM-SDHI

$$x \leftarrow_{\$} Z_p; \forall t, q_t \leftarrow \theta$$

$$y_1, \dots, y_m \leftarrow_{\$} (Z_p)^m$$

$$C_1, \dots, C_l, t \leftarrow_{\$} A^{SDH}(g^x, g^{y_1}, \dots, g^{y_m})$$

A wins if $q_t < l \wedge \forall i, \exists j, C_i = y_j^{1/(x+t)}$

- Relation to POPRF security: Per-tag query counter ensures each blind evaluation results in the client learning at most one PRF evaluation restricted to the queried tag
- Reduction to uber assumption: Argue linear independence of elements given to adversary and winning elements
- Challenge: Repeated queries to SDH that “mix” tags

Oracle SDH(B, t)

$$q_t \leftarrow q_t + 1$$

Return $B^{1/(x+t)}$

Per-tag query counter!

One-more (Gap) Strong Diffie-Hellman Inversion

OM-SDHI

$$x \leftarrow_{\$} Z_p; \forall t, q_t \leftarrow 0$$

$$y_1, \dots, y_m \leftarrow_{\$} (Z_p)^m$$

$$C_1, \dots, C_l, t \leftarrow_{\$} A^{SDH}(g^x, g^{y_1}, \dots, g^{y_m})$$

A wins if $q_t < l \wedge \forall i, \exists j, C_i = y_j^{1/(x+t)}$

Oracle SDH(B, t)

$$q_t \leftarrow q_t + 1$$

Return $B^{1/(x+t)}$

Per-tag query counter!

Leads to non-trivial polynomial independence argument!

- Relation to POPRF evaluation results in the client learning at most one PRF evaluation restricted to the queried tag
- Reduction to uber assumption: Argue linear independence of elements given to adversary and winning elements
- Challenge: Repeated queries to SDH that “mix” tags

Crux: Handling “mixed” SDHI queries

Oracle SDH(B, t)

$q_t \leftarrow q_t + 1$
Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$		
input tag	t_1		
output	$R_1 = g^{\frac{y_1}{x+t_1}}$		

Crux: Handling “mixed” SDHI queries

Oracle $SDH(B, t)$

$q_t \leftarrow q_{t+1}$
Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$	$B_2 = g^{y_2}$	
input tag	t_1	t_2	
output	$R_1 = g^{\frac{y_1}{x+t_1}}$	$R_2 = g^{\frac{y_2}{x+t_2}}$	

Crux: Handling “mixed” SDHI queries

Oracle $SDH(B, t)$

 $q_t \leftarrow q_{t+1}$
 Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$	$B_2 = g^{y_2}$	$B_3 = R_1 g^{y_3} = g^{y_3 + \frac{y_1}{x+t_1}}$
input tag	t_1	t_2	t_1
output	$R_1 = g^{\frac{y_1}{x+t_1}}$	$R_2 = g^{\frac{y_2}{x+t_2}}$	

Repeated query to t_1 but does not mix tags!

Crux: Handling “mixed” SDHI queries

Oracle $SDH(B, t)$

$q_t \leftarrow q_{t+1}$

Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$	$B_2 = g^{y_2}$	$B_3 = R_1 g^{y_3} = g^{y_3 + \frac{y_1}{x+t_1}}$
input tag	t_1	t_2	t_1
output	$R_1 = g^{\frac{y_1}{x+t_1}}$	$R_2 = g^{\frac{y_2}{x+t_2}}$	$R_3 = g^{\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}}$

Repeated query to t_1 but does not mix tags!

Crux: Handling “mixed” SDHI queries

Oracle $SDH(B, t)$

$q_t \leftarrow q_{t+1}$
Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$	$B_2 = g^{y_2}$	
input tag	t_1	t_2	
output	$R_1 = g^{\frac{y_1}{x+t_1}}$	$R_2 = g^{\frac{y_2}{x+t_2}}$	

Crux: Handling “mixed” SDHI queries

Oracle $SDH(B, t)$

$q_t \leftarrow q_t + 1$

Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$	$B_2 = g^{y_2}$	$B_3 = R_1 R_2 g^{y_3} = g^{y_3 + \frac{y_1}{x+t_1} + \frac{y_2}{x+t_2}}$
input tag	t_1	t_2	t_1
output	$R_1 = g^{\frac{y_1}{x+t_1}}$	$R_2 = g^{\frac{y_2}{x+t_2}}$	

Repeated query to t_1 that mixes tags!

Crux: Handling “mixed” SDHI queries

Oracle $SDH(B, t)$

$q_t \leftarrow q_t + 1$
Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$	$B_2 = g^{y_2}$	$B_3 = R_1 R_2 g^{y_3} = g^{y_3 + \frac{y_1}{x+t_1} + \frac{y_2}{x+t_2}}$
input tag	t_1	t_2	t_1
output	$R_1 = g^{\frac{y_1}{x+t_1}}$	$R_2 = g^{\frac{y_2}{x+t_2}}$	$R_3 = g^{\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}}$

Repeated query to t_1 that mixes tags!

Crux: Handling “mixed” SDHI queries

Oracle $SDH(B, t)$
 $q_t \leftarrow q_{t+1}$
 Return $B^{1/(x+t)}$

	query 1	query 2	query 3
input element	$B_1 = g^{y_1}$	$B_2 = g^{y_2}$	$B_3 = R_1 R_2 g^{y_3} = g^{y_3 + \frac{y_1}{x+t_1} + \frac{y_2}{x+t_2}}$
input tag	t_1	t_2	t_1
output	$R_1 = g^{\frac{y_1}{x+t_1}}$	$R_2 = g^{\frac{y_2}{x+t_2}}$	$R_3 = g^{\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}}$

Repeated query to t_1 that mixes tags!

In reduction to uber assumption, need to show linear independence of exponents

Crux: Handling “mixed” SDHI queries

query 1 query 2

query 3

independence
argument

t_1 t_2 t_3

Crux: Handling “mixed” SDHI queries

query 1

query 2

query 3

independence
argument

$t_1 t_2 t_3$

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

Crux: Handling “mixed” SDHI queries

query 1

query 2

query 3

independence
argument

t_1 t_2 t_3

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

Crux: Handling “mixed” SDHI queries

query 1

query 2

query 3

independence
argument

t_1 t_2 t_3

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

??

? ? 0

Crux: Handling “mixed” SDHI queries

query 1

query 2

query 3

independence
argument

t_1 t_2 t_3

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

??

?? ?? 0

Idea: “Unmix” mixed query while preserving span

query 1

query 2

query 3

independence
argument

$t_1 \ t_2 \ t_3$

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

??

?? ?? 0

Idea: “Unmix” mixed query while preserving span

query 1

query 2

query 3

independence
argument

$t_1 \ t_2 \ t_3$

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

??

?? ?? 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{\alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

Idea: “Unmix” mixed query while preserving span

query 1

query 2

query 3

independence
argument

t_1 t_2 t_3

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

$$\frac{y_1}{x+t_1} \quad \frac{y_2}{x+t_2} \quad \frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

??

?? ?? 0

$$\frac{y_1}{x+t_1} \quad \frac{y_2}{x+t_2} \quad \frac{\alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

Same span!

Idea: “Unmix” mixed query while preserving span

query 1

query 2

query 3

independence
argument

t_1 t_2 t_3

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

??

?? ?? 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{\alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

Idea: “Unmix” mixed query while preserving span

query 1

query 2

query 3

independence
argument

t_1 t_2 t_3

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

unmixing!

2 1 0

$$\frac{y_1}{x+t_1}$$

$$\frac{y_2}{x+t_2}$$

$$\frac{\alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

Idea: “Unmix” mixed query while preserving span

Main Lemma (informal):

A transcript of $n+1$ queries where the first n queries are “unmixed” can be rewritten as a transcript of $n+1$ queries where the $n+1^{\text{th}}$ query q_{n+1} is replaced with “unmixed” query q such that the span of the two transcripts are the equivalent.

$$\frac{y_1}{x+t_1} \quad \frac{y_2}{x+t_2} \quad \frac{y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2} + \frac{y_2}{(x+t_1)(x+t_2)}$$

unmixing!

2 1 0

$$\frac{y_1}{x+t_1} \quad \frac{y_2}{x+t_2} \quad \frac{\alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_3}{x+t_1} + \frac{y_1}{(x+t_1)^2}$$

easy

2 1 0

Impact

- Implementation confirms practicality
 - 25% overhead over 2HashDH (~ 400μs vs 500μs)
- Interest by companies deploying OPRF applications
- Incorporated into OPRF standardization effort
 - <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-voprf>

Conclusion

- Motivation: POPRFs simplify use of OPRFs in existing applications
- 3HashSDHI: A new, simple POPRF
 - Previous candidate POPRFs rely on pairings or RSA groups
 - Shares similar structure to standards-track 2HashDH OPRF
- Non-trivial security analysis: OM-Gap-SDHI
 - Recovers security of closely related partially blind signature scheme

Standards track: <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-voprf>

Archive: <https://ia.cr/2021/864>