Verifiable Oblivious Pseudorandom Functions

draft-sullivan-cfrg-voprf

Nick Sullivan (nick@cloudflare.com)
Christopher A. Wood (cawood@apple.com)

CFRG
IETF 101, March 2018, London

VRFs

Public-key versions of cryptographic hash functions [1]

- Signers compute the “hash,” and verifiers check its correctness

Applications

- Key transparency [https://github.com/google/keytransparency]
- NSEC5 [https://tools.ietf.org/html/draft-vcelak-nsec5-06]

[1] https://datatracker.ietf.org/doc/draft-irtf-cfrg-vrf/

OPRFs

Related to blind signatures rooted in e-cash schemes [1]

- Verifier requests signature over blinded message
- Signer signs blinded message
- Verifier removes blind from signature and message, yielding a message with valid signature

Obliviousness: Signer learns nothing of message to sign

[1] Blind signatures for untraceable payments

VOPRFs

Goal: Make OPRFs verifiable

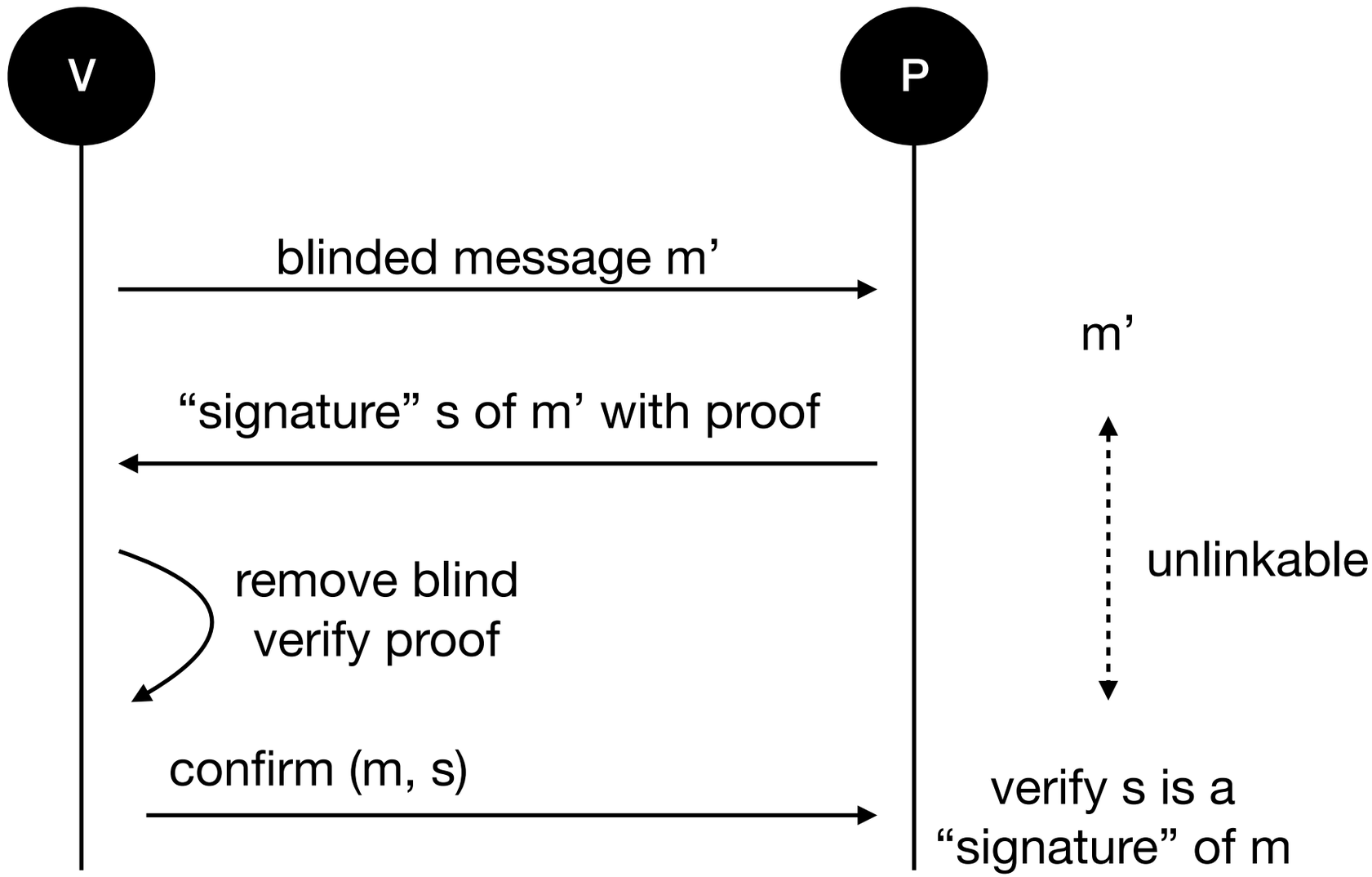
PRF criteria:

- Given x, it is infeasible to compute y = F(k, x) without k
- F is indistinguishable from a truly random function

Verifiable: Verifier can verify that Y = kG was used to compute y = F(k, x)

Oblivious: Signer does not know which signing operation was used in the computation of y

VOPRFs


V
P
blinded message m’
m’
“signature” s of m’ with proof
unlinkable
remove blind
verify proof
confirm (m, s)
verify s is a
“signature” of m

EC-VOPRF

A VOPRF can be constructed with from the following building blocks:

- An elliptic curve
- A one-way function to hash a value to a point on a prime order subgroup of the elliptic curve (H2C)
- A non-interactive zero knowledge discrete log equality proof algorithm for points in this subgroup (DLEQ)

NIZK Discrete Log Equality

$$\log_G(Y) =? \log_M(Z)$$
$$(Y = kG, Z = kM)$$

DLEQGenerate(G, Y, M, Z)	DLEQVerify$(G, Y, M, Z, (c, s))$
$r \leftarrow \mathbb{Z}_p$	$A' = sG + cY$
$A = rG$	$B' = sM + cZ$
$B = rM$	$c' = H(G, Y, Z, A', B')$
$c = H(G, Y, Z, A, B)$	Output $c == c'$
$s = (r - ck)(\mathrm{mod} p)$	
Output (c, s)	

Jarecki, Kiayias, Krawczyk. Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only model.

Protocol

ECVOPRF Protocol

V		P
$r \leftarrow\$ \{\mathbb{Z}_p\}$		
$M = r\mathsf{H2C}(x)$		
	$\xrightarrow{M}$	
		$Z = kM = kr\mathsf{H2C}(x)$
		$(c, s) = \mathsf{DLEQGenerate}(G, Y, M, Z)$
	$\xleftarrow{(Z, c, s)}$	
$b = \mathsf{DLEQVerify}(G, Y, M, Z, (c, s))$		
$y = r^{-1}Z = k\mathsf{H2C}(x)$		
$\mathsf{If} b = 1, \mathsf{Output}(H(y)), \mathsf{ElseOutput}(\perp)$		

Note: batched protocol is possible with single DLEQVerify

Confirmation by Prover

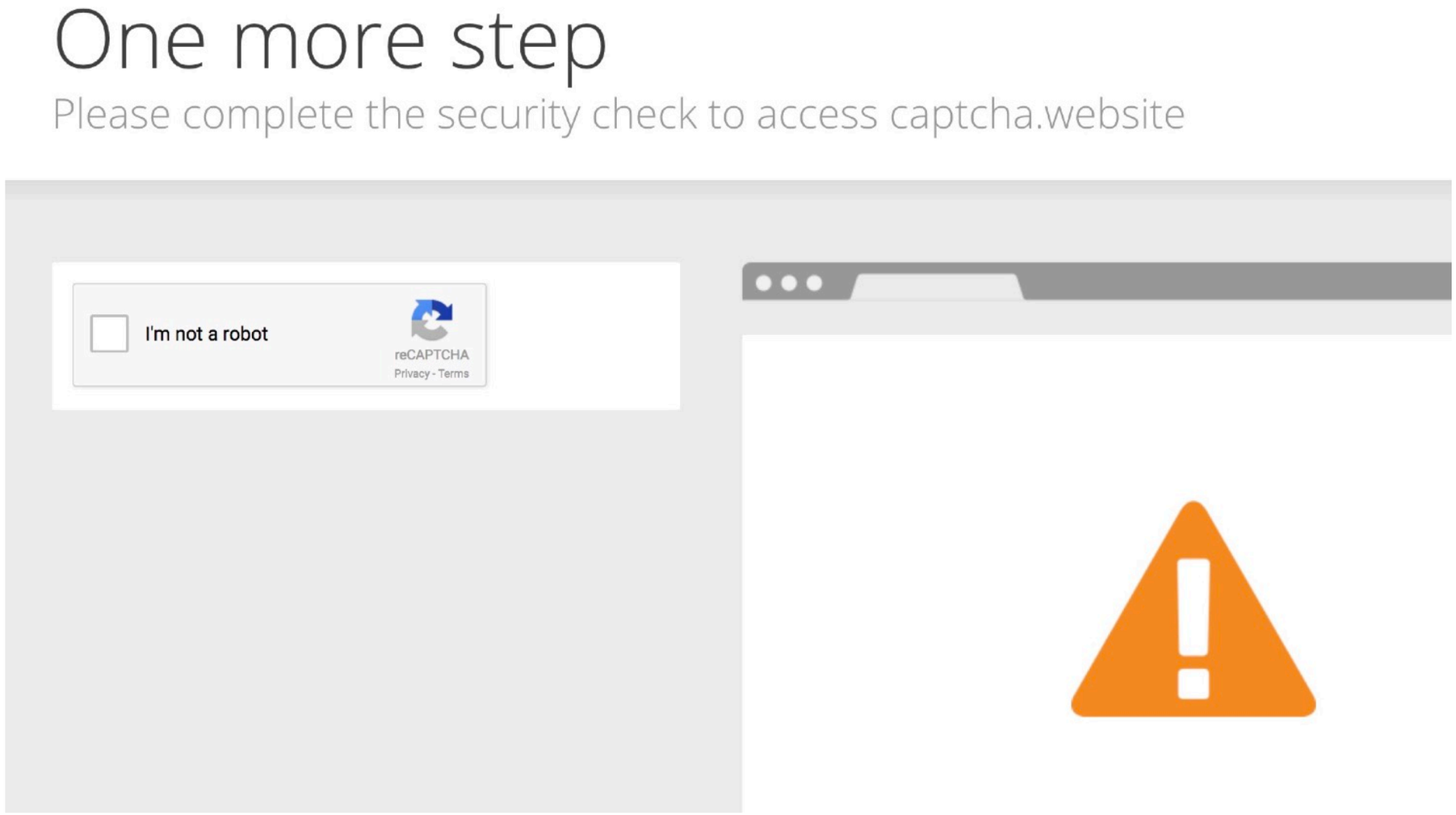
(x, y) can be sent to P

P computes kH2C(x) and checks kH2C(x) =? y

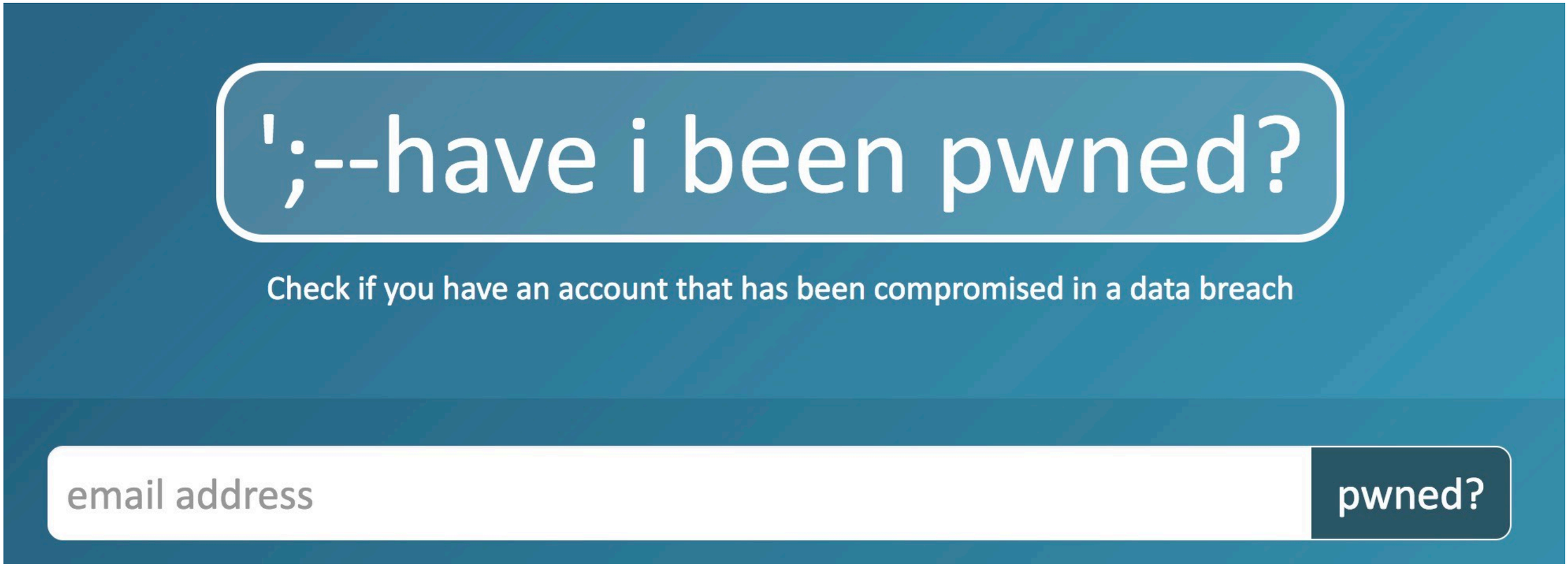
If these match, P can confirm that x was used in a previous VOPRF prover request

Alternatively, (x, m, PRF(y, m)) can be used for a binding message m

Application: Privacy Pass



Application: Privacy-Preserving Password Leak Checks



Foundations

Jarecki et al., “Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model.” [https://eprint.iacr.org/2014/650.pdf]

Pedersen, “Wallet Databases with Observers.” [https://link.springer.com/content/pdf/10.1007/3-540-48071-4_7.pdf]

Davidson et al., “Privacy Pass: Bypassing Internet Challenges Anonymously” PETS 2018, forthcoming.

Sullivan et al., “Hashing to Elliptic Curves.” [draft-sullivan-cfrg-hash-to-curve]