OPAQUE: Strong client-server password authentication for standardization

Hugo Krawczyk
IBM Research

RWC’2019 – 01-09-19

bit.ly/OPAQUE-paper: S. Jarecki, H. Krawczyk, J. Xu, Eurocrypt 2018

bit.ly/OPAQUE-draft: draft-krawczyk-cfrg-opaque-01

Jiayu Xu

JiayuX@uci.edu

Passwords

Passwords, can't live with them, can't live without them

Cyberius Erasmus

- If you are one of those that believe passwords are about to disappear, this talk is not for you
  - □ I was in that camp 25 years ago... life taught me I was wrong
  - □ Deployment, convenience, portability , familiarity, inertia, …

Unavoidable attacks

- Online guessing
  - Mitigation: throttling, second factor
- Offline dictionary search: *Upon server compromise*
  - Mitigation: Salting! E.g., server stores pairs ($salt_U$, Hash($salt_U$, pwd_U))
  - Make sure the exhaustive attack starts *after* the compromise happens (no pre-computed dictionaries)

Avoidable attacks

Plaintext-password visibility: Weakness of *password-over-TLS*

- Password visible at the server, upon TLS decryption

  (in particular, vulnerable to insiders, debugging tools, etc.)

- In transit (PKI failures):
  - □ TLS failures: implementation/misconfig, certificates, user mishandling, …
  - □ By design: Middle boxes (CDN, monitoring, security, …)

- Phishing that exploits visibility of password at server

Pre-computed dictionaries: Offline attack unavoidable upon server compromise, but no pre-computation should help (essential salt)

aPAKE

- aPAKE: Asymmetric Password Authenticated Key Exchange ('a' also for 'augmented')
- Asymmetric: Client-Server setting
  - User has a password pwd, server has a one-way mapping of pwd
  - Contrast with symmetric case where both peers store the same password
- No pre-computation attacks
  - Unavoidable offline dictionary attack but only upon server compromise
- Plaintext password never visible to server
- PKI free: "password only"

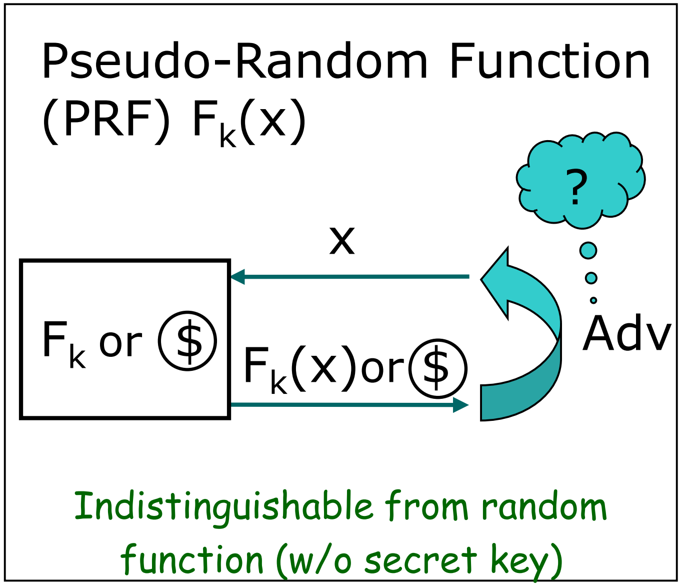
OPAQUE: An Asymmetric PAKE

- First aPAKE secure against pre-computation attacks:
  - □ All other aPAKE protocols don't use salt or transmit it in the clear!
    → targeted dictionaries
    - True even for proven protocols (weak model)
    - SPAKE2+, AugPAKE, SRP, etc. (no proof, even in a weak model)
- PKI-free (user only remembers its password)

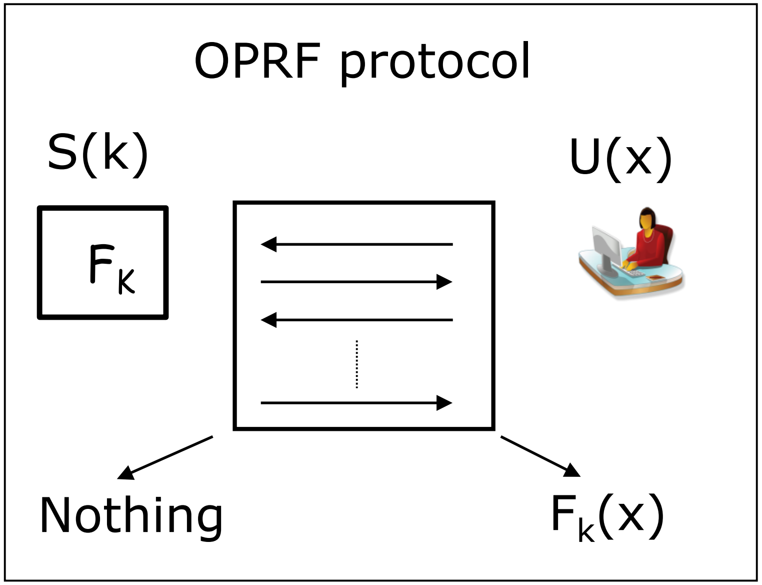
  Note: secure channels needed only during password registration
- Password *never* in the clear outside client domain (not even at registr.)
- More to like. But first…

The OPAQUE Protocol

Oblivious PRF (OPRF)



Indistinguishable from random function (w/o secret key)



- ❑ OPRF: Protocol b/w a user with input x and server with key k; user learns $F_k(x)$ and *nothing else* and server learns *nothing* (neither the input or output of the computation)

OPAQUE: Basic idea

[FK'00, Boyen'09 (HPAKE) , JKKX'16]

- U runs OPRF with S by which it "exchanges" its password pwd for the pseudo-random OPRF output $\boxed{rwd = OPRF_K(pwd)}$
- S (or anyone else) learns *nothing* about pwd and rwd
  → rwd is a *strong crypto key* for anyone that does not know pwd
- U uses rwd as a private key in a key exchange (KE) protocol with S
- OPAQUE (assume public-key KE w/ keys ($priv_U$,pub_U,$priv_S$,pub_S))
  - □ At registration U stores at S: Env_U = $AuthEnc_{rwd}$($priv_U$, pub_S);
    S also stores OPRF key k, privS, pubS, pub_U .
  - □ For login: U and S run $OPRF_K(pwd)$, U decrypts Env_U and runs KE with S

➔ OPAQUE: “compiler” from <u>any</u> OPRF and KE* to aPAKE

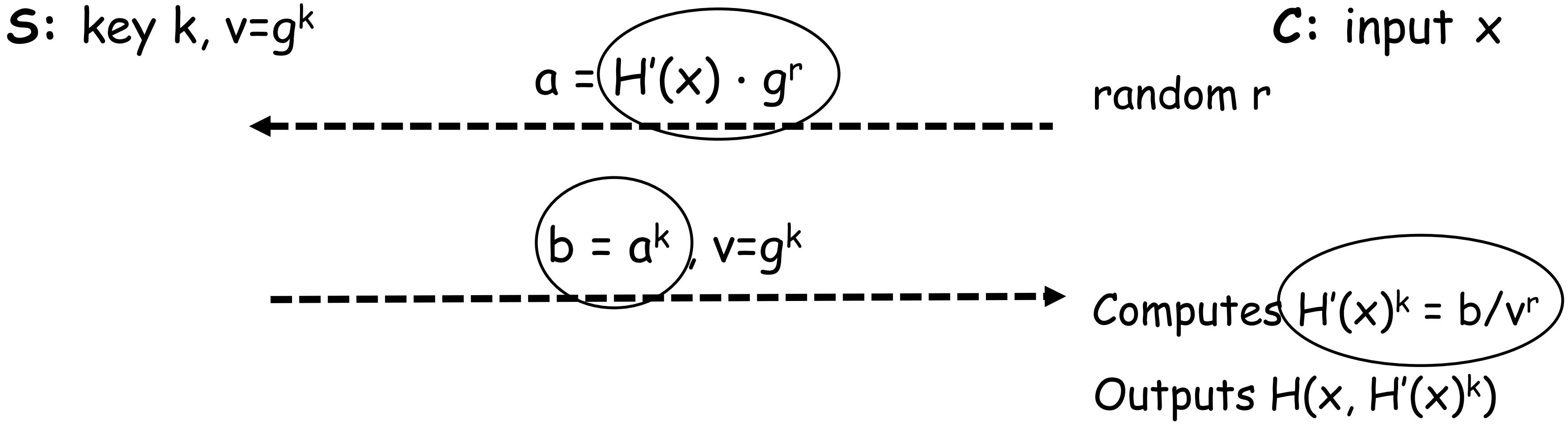
* KE with the KCI property (security against “reverse impersonation”)

DH-OPRF

- **PRF**: $F_K(x) = H(x, H'(x)^k)$ over group G with generator g;

  key =random exponent k; H' hashes x into a random element in G.

- Oblivious computation via Blind DH Computation (C has x, S has k)



S: key k, $v=g^k$

C: input x

random r

$a = H'(x) \cdot g^r$

$b = a^k$, $v=g^k$

Computes $H'(x)^k = b/v^r$

Outputs $H(x, H'(x)^k)$

- The blinding factor g^r works as a one-time encryption key,

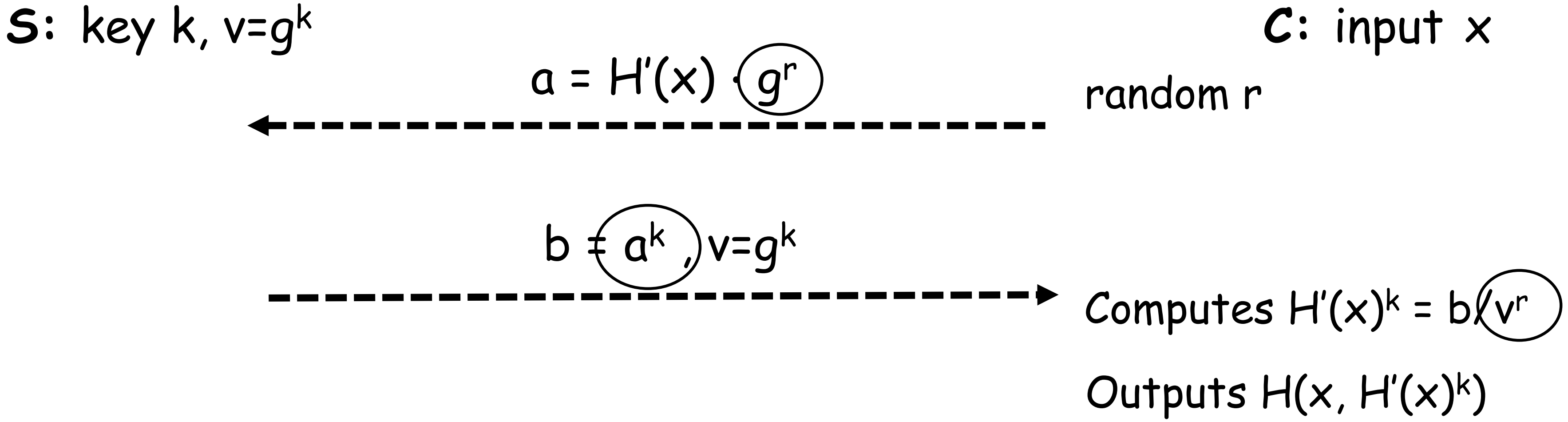
  hence ***it hides*** H'(x) ***and*** x ***perfectly*** (from S)

Note: OPAQUE includes v under H

DH-OPRF

- **PRF**: $F_K(x) = H(x, H'(x)^k)$ over group G with generator g; key =random exponent k; H' hashes x into a random element in G.

- Oblivious computation via Blind DH Computation (C has x, S has k)



S: key k, $v=g^k$

C: input x

random r

$a = H'(x) \cdot g^r$

$b = a^k$, $v=g^k$

Computes $H'(x)^k = b/v^r$

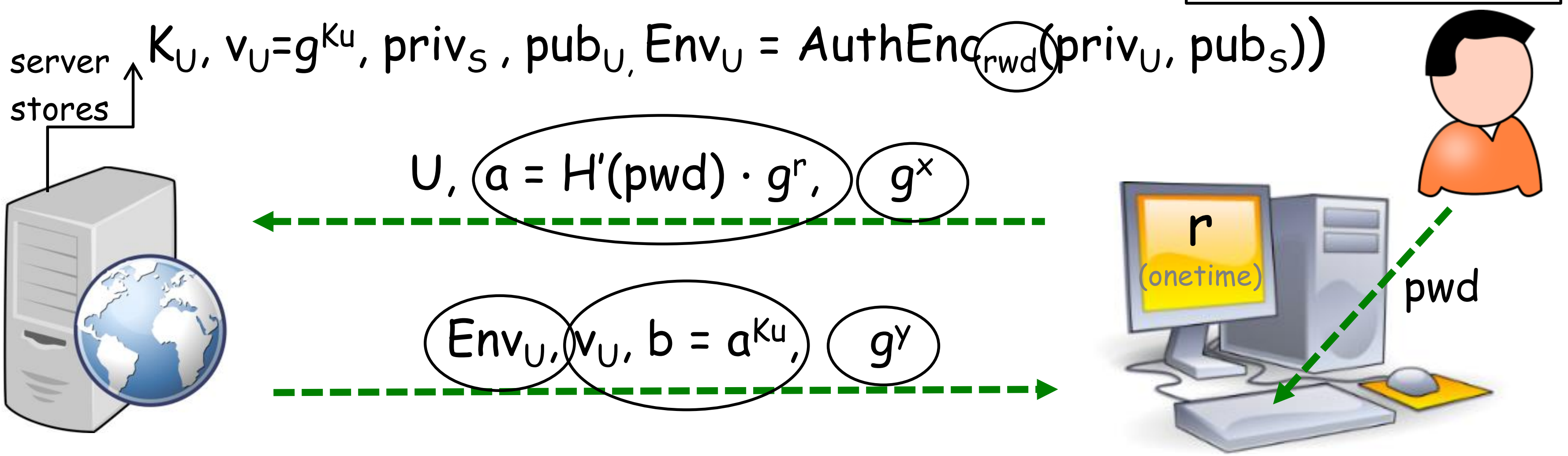
Outputs $H(x, H'(x)^k)$

- Communication: Single round

  Computation: 1 exponentiation for server, 2 for client (of which one or two can be fixed base), hash-into-curve op for C

OPAQUE with DH-OPRF

rwd = $\mathrm{OPRF}_K(\mathrm{pwd})$



server stores K_U, $v_U = g^{Ku}$, priv_S , $\mathrm{pub}_{U,}$ $\mathrm{Env}_U = \mathrm{AuthEnc}_{\mathrm{rwd}}(\mathrm{priv}_U, \mathrm{pub}_S))$

U, $a = H'(\mathrm{pwd}) \cdot g^r$, g^x

Env_U, v_U, $b = a^{Ku}$, g^y

C: rwd = $H(\mathrm{pwd}, b/v_U{}^r)$; $\mathrm{priv}_U, \mathrm{pub}_S \leftarrow \mathrm{Dec}_{\mathrm{rwd}}(\mathrm{Env}_U)$;

. $SK = KE(\mathrm{priv}_U, x, \mathrm{pub}_S, g^y)$

S: $SK = KE(\mathrm{priv}_S, y, \mathrm{pub}_U, g^x)$

- Example: DH-OPRF + KE=HMQV

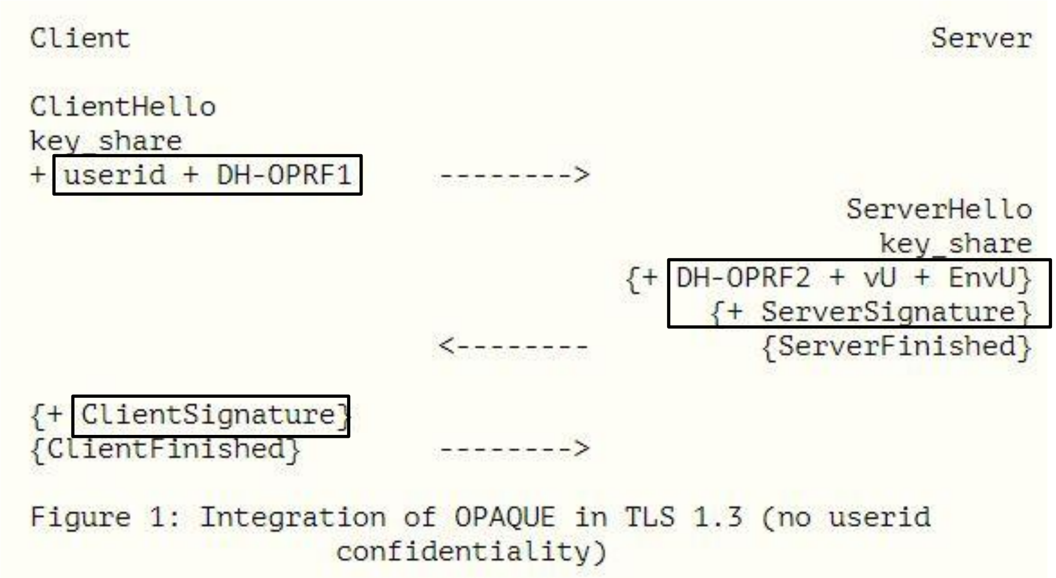
Single round + total cost of ~ 2.5 var-base exponentiations for C and S and a hash-to-curve operation for C

Instantiations

- OPAQUE with HMQV
  - Single round, about 2.5 exponentiations for server and client (additional message with a MAC from C to S for explicit authentication)
  - Can use other implicit authenticated protocols w/ additional exponentiation (e.g. signal/X3DH, NAXOS, etc.)
- OPAQUE with SIGMA
  - Adds a signature from server to client and an additional message from client with its signature (signatures under $priv_S$ and $priv_U$, respectively)

OPAQUE with TLS

- Blends smoothly with 3-flight TLS 1.3 handshake
  - server’s cert replaced with $priv_S$ and client’s signature uses $priv_U$



```
Client                                               Server

ClientHello
key_share
+ userid + DH-OPRF1     -------->
                                                ServerHello
                                                  key_share
                                   {+ DH-OPRF2 + vU + EnvU}
                                        {+ ServerSignature}
                        <--------          {ServerFinished}

{+ ClientSignature}
{ClientFinished}        -------->
```

Figure 1: Integration of OPAQUE in TLS 1.3 (no userid confidentiality)

- For user account confidentiality: Adds a round trip, with account info protected by a server-authenticated 1-RTT (with server’s cert)

OPAQUE with TLS

Hedging TLS:

- From “TLS-protected passwords” to “password-protected TLS”
- Security as long as PKI and/or password are secure

ask not what TLS can do for passwords

- ask what passwords can do for TLS

OPAQUE Security

- Proven secure against pre-computation attacks !!
  (OPRF key k acts as “secret salt”)
- Proof (UC model):
  - Strong aPAKE model (PKI-free and disallows pre-computation attacks)
  - Proof of OPAQUE is generic: OPRF + KE (w/ KCI) + Key-robust AEnc
  - With DH-OPRF: In ROM under Gap One-More Diffie-Hellman
- Forward security (crucial if password eventually leaks)
- User-side hash iterations (e.g., PBKDF2, scrypt, aragon2)
  - increased security against offline attacks upon server compromise

Extensions

- Credential retrieval:
  - □ Env_U can include additional secret/authenticated information
- Multi-server implementation
  - □ Threshold OPRF [JKKX'17 eprint.iacr.org/2017/363]
  - □ Attacker needs to break into a threshold number of servers
  - □ Even then it can only mount a dictionary attack
  - □ User/client transparent: User need not be aware of the distributed implementation (communicates via gateway)

Summary: OPAQUE Protocol

- Modular/flexible: Can compose with any Authenticated KE (w/KCI)
- Efficient instantiations (e.g., HMQV, SIGMA, TLS 1.3)
- Smooth integration with TLS
  - Much stronger than current password-over-TLS
  - Hedging against PKI failures: "password-protected TLS"
- Extensions:
  - Credential retrieval
  - user-transparent multi-server implementation (threshold security)

Summary: OPAQUE Security

- Secure against pre-computation attacks (first *true* aPAKE)
- Password *never* in the clear outside client domain
- No reliance on PKI
- Forward secure (critical for when password leaks)
- Client-side hardening (e.g. iterated hashes, scrypt, etc.)
- Proof in a strong UC security model

Standardization

- IF we are looking for a strong aPAKE to standardize (are we?) OPAQUE seems to fit perfectly
  - True aPAKE security, modular, efficient, extra properties
- In particular, a good fit for TLS 1.3
  - From TLS-protected-password to password-protected-TLS
- CFRG and TLS working groups

Thanks!

- bit.ly/OPAQUE-paper:
  - S. Jarecki, H. Krawczyk, J. Xu, Eurocrypt 2018
- bit.ly/OPAQUE-draft:
  - draft-krawczyk-cfrg-opaque-01
- bit.ly/SPHINX-paper - password manager to complement OPAQUE
  - SPHINX: A Password Store that Perfectly Hides Passwords from Itself
  - RWC’2017