# OPAQUE: A Strong Asymmetric PAKE Protocol Secure Against Pre-Computation Attacks

Stanislaw Jarecki, Hugo Krawczyk, **Jiayu Xu**

# Motivation: Password Authentication

- Passwords are the prevalent tool for authentication
- Passwords are vulnerable to various attacks
  - Human memorable $\Rightarrow$ low-entropy
  - Reusing the same / highly correlated password

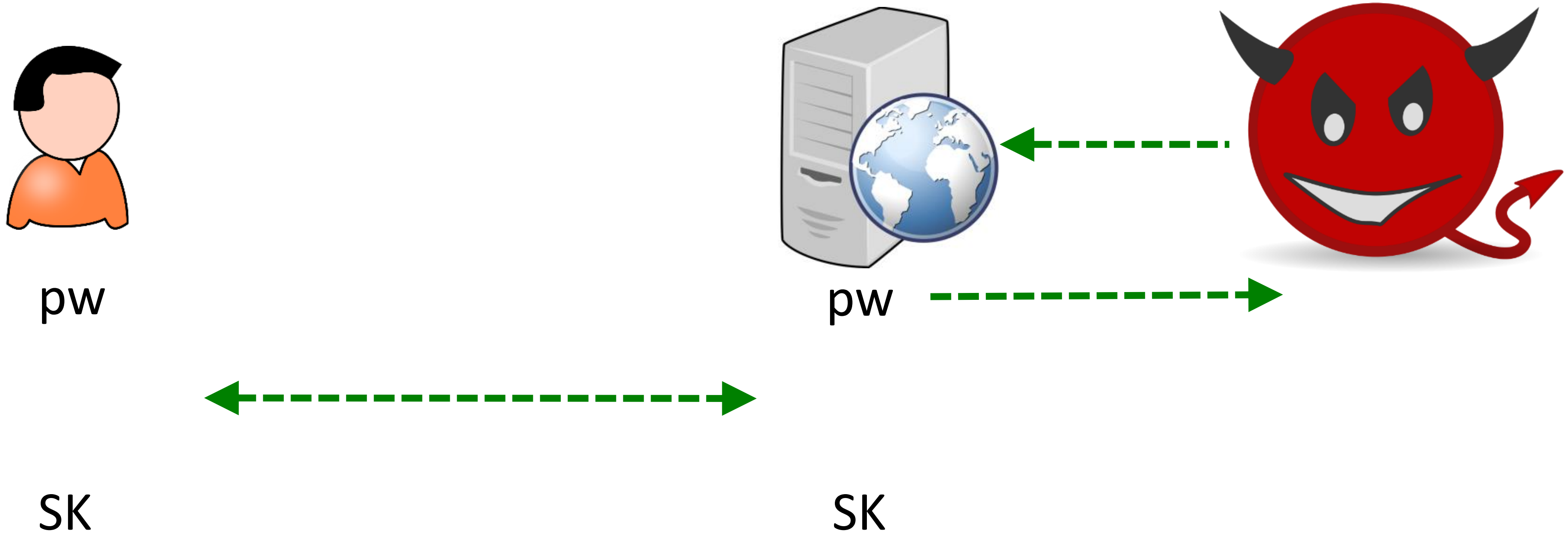
# Password Protocols in Crypto Literature

- (Symmetric) Password-Authenticated Key Exchange (PAKE) [BMP’00, BPR’00]


- Password-only: no Public Key Infrastructure (PKI)!

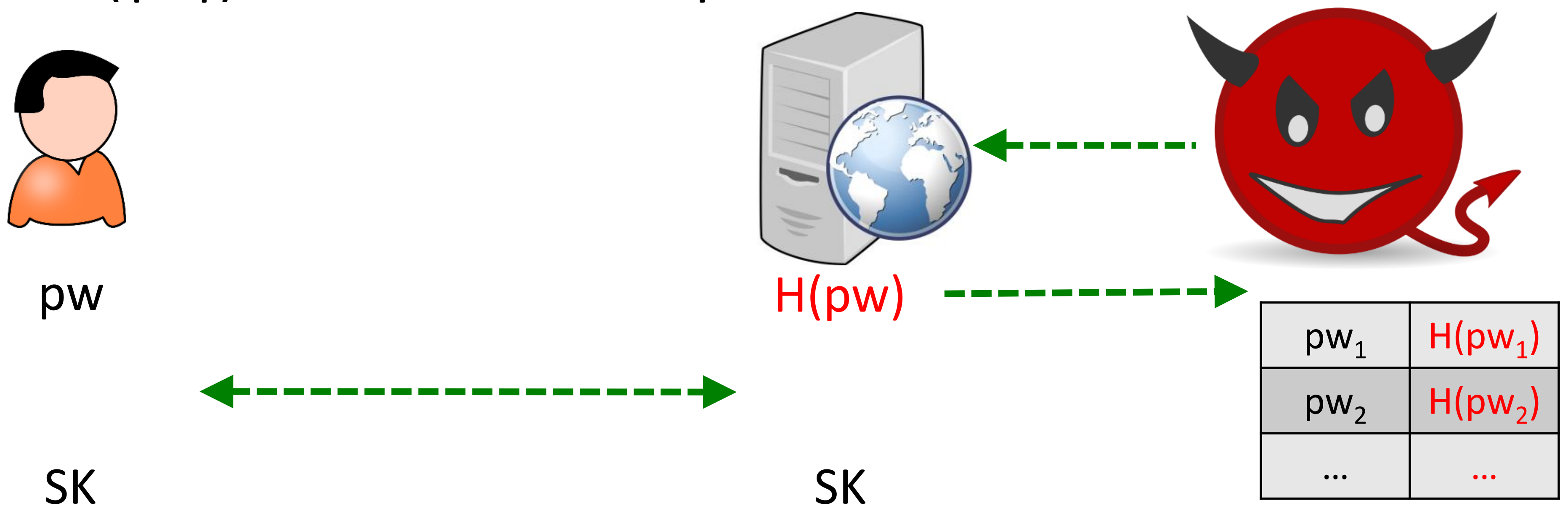
# PAKE in the Client-Server Setting…

- Server compromised ⇒ password leaked straight away!



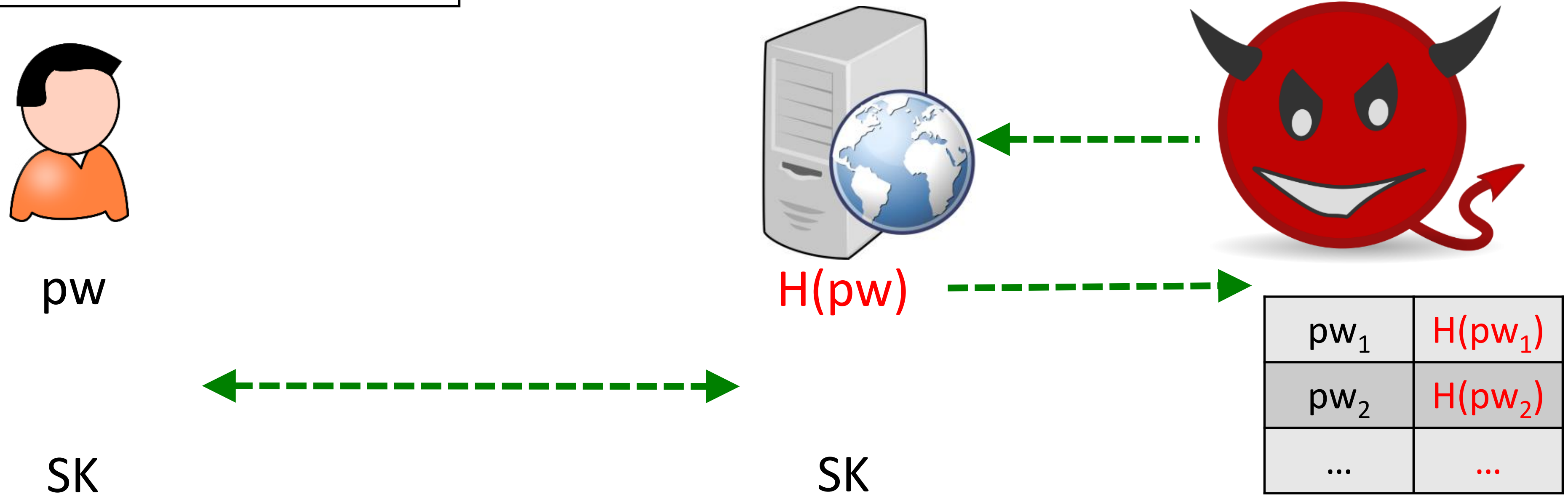
# Asymmetric / Augmented PAKE (aPAKE) [BM'93, BMP'00, GMR'06]

- Server stores a mapping of the password ("password file")
- Server compromised ⇒ only unavoidable offline dictionary attack allowed ⇒ O(|D|) time to learn the password



# Wait, What if the Adversary…

- …computes the hash table prior to compromising the server…
  - …and upon compromising the server, compares the password file against the pre-computed hash values?
- **“pre-computation attack”**



# Pre-Computation Attack

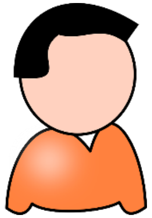
- $O(\log|D|)$ time to learn the password after server compromise!
- How to force the adversary to spend $O(|D|)$ time on offline dictionary attack after server compromise?
  - Store (s,H(pw,s)) where s is a private random salt

- Strong aPAKE (SaPAKE): secure against pre-computation attacks

# aPAKE: State-of-Art

- Formal definition
  - Game-based [BMP'00, BP'13]
  - Universally-composable (UC) [GMR'06]
- Very few proposals proven secure
- **All of them allow for pre-computation attack!**
  - No salt in password hash / salt is sent in the clear
  - Does not quite meet the motivation behind aPAKE definition…

# In Practice: Password-over-TLS



pw

(s,H(pw,s))

TLS(pw)

pw

check against password file

# Password-over-TLS v. aPAKE

| Password-over-TLS | aPAKE |
| --- | --- |
| Requires PKI | Password-only |
| Server sees password upon TLS decryption | Server never sees password |
| Requires full offline dictionary attack upon server compromise | Allows for pre-computation attack |

- Strong aPAKE: combines the good parts of both!

# Our Contributions

- (1) The first definition of Strong aPAKE
- …and it is in the UC setting
  - Preferable than game-based definitions (non-uniform distribution of password, password correlation, easier to argue, etc.)
- (2) Two highly efficient realizations of Strong aPAKE (the latter named OPAQUE) in the Random Oracle Model (ROM)
- …and proven secure in the UC setting

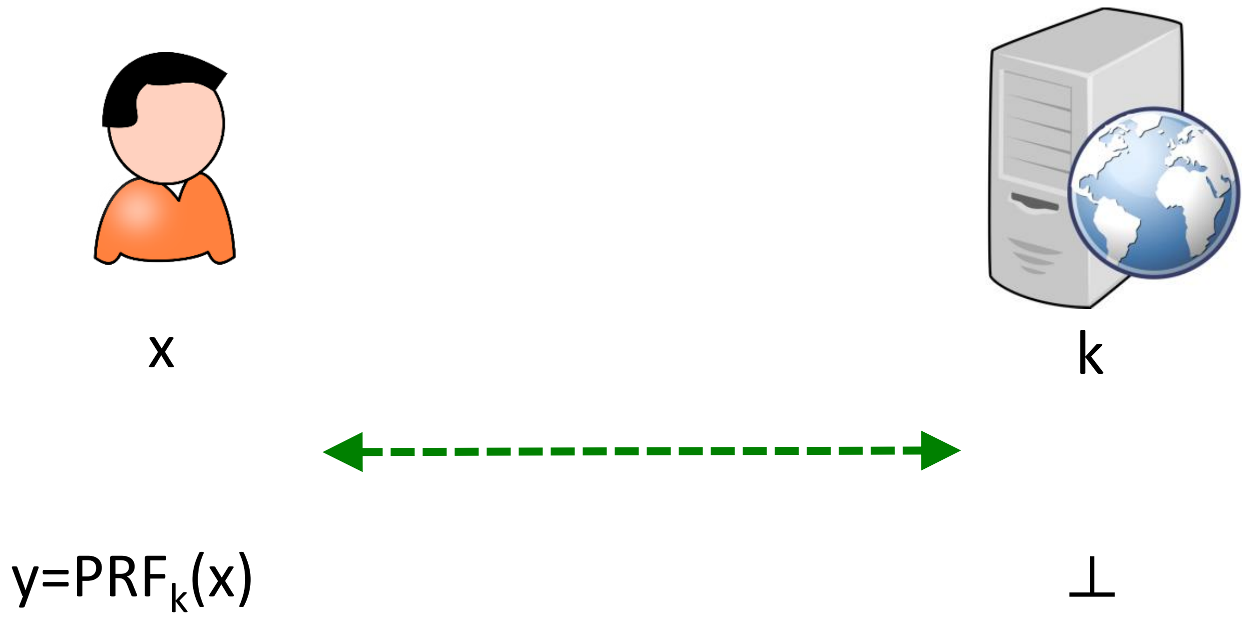
- The UC aPAKE functionality in [GMR'06] (full text)
  - …Allows for pre-computation attack (grey text)
- Our UC SaPAKE functionality does not (grey text omitted)

Stealing Password Data

– On (StealPwdFile, *sid*) from $\mathcal{A}^*$, if there is no record (file, U, S, pw), return "no password file" to $\mathcal{A}^*$. Otherwise, if the record is marked uncompromised, mark it compromised; regardless,
  - If there is a record (offline, pw), send pw to $\mathcal{A}^*$.
  - Else Return "password file stolen" to $\mathcal{A}^*$.

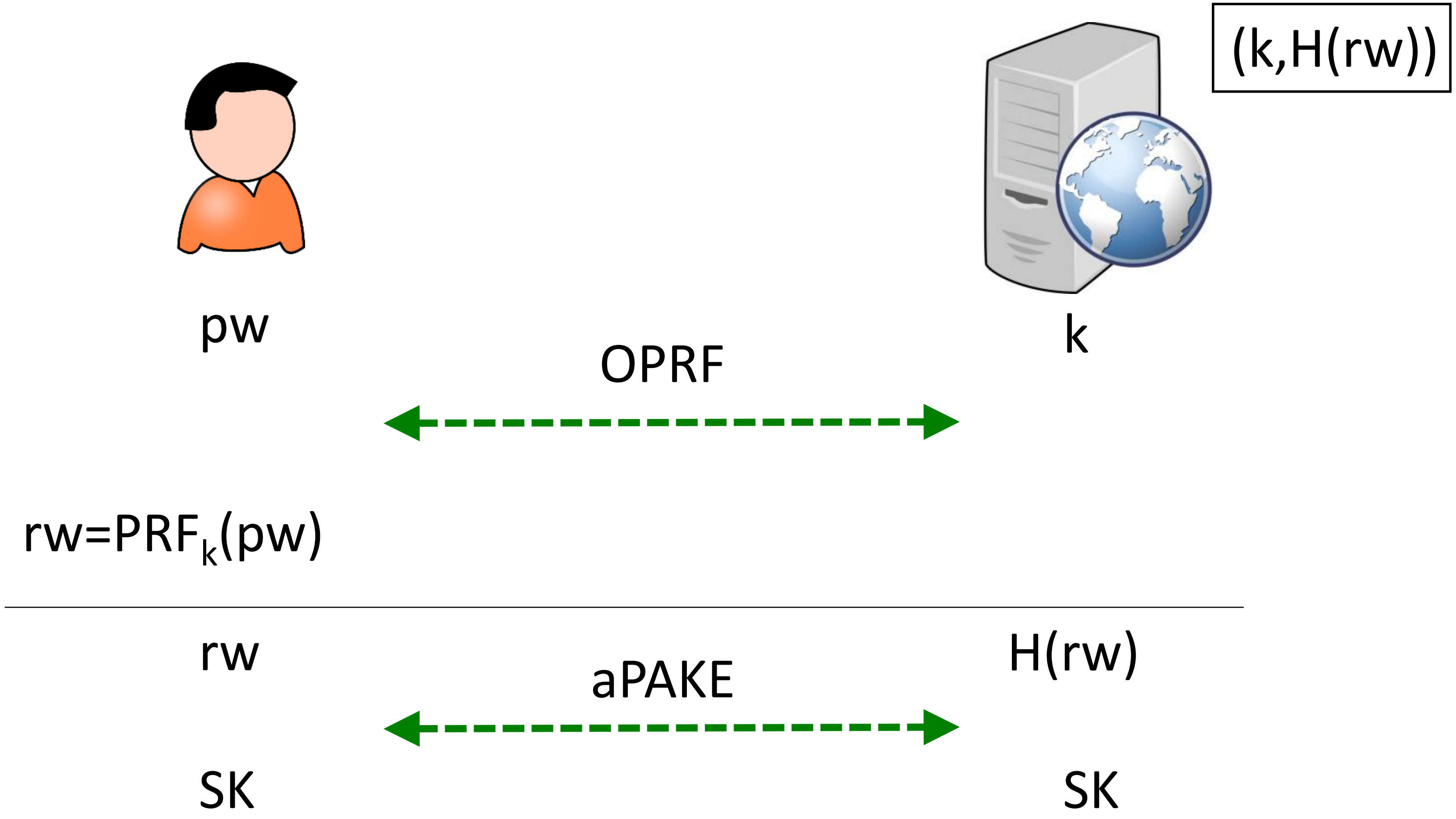
– On (OfflineTestPwd, *sid*, pw$^*$) from $\mathcal{A}^*$, do:
  - If there is a record (file, U, S, pw) marked compromised, do: if pw$^*$ = pw, return "correct guess" to $\mathcal{A}^*$; else return "wrong guess."
  - Else record (offline, pw).

# Our Tool: Oblivious PRF (OPRF) [NR’97, FIPR’05, JKK’14]



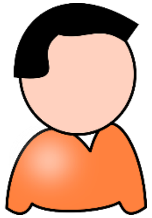

- Very efficient instantiation: DH-OPRF (in the UC setting [JKKX’16])

# Construction #1: OPRF + aPAKE → SaPAKE




- rw is random to the adversary ⇒ cannot launch pre-computation attack on rw (thanks to k)

# Construction #2: OPRF + AKE → SaPAKE



$(k, c, priv_S, pub_S, pub_U)$

pw

k

OPRF

$rw = PRF_k(pw)$

$c = AuthEnc_{rw}(priv_U, pub_U, pub_S)$

$priv_U, pub_S, pub_U$

$priv_S, pub_S, pub_U$

AKE

SK

SK

* AKE has the Key Compromise Impersonation (KCI) property

# OPAQUE

- Practical instantiation of “OPRF+AKE” construction
  - Very efficient (based on DH-OPRF)
  - AKE can be instantiated using various protocols
- Variants studied previously [FK’00, Boyen’09, JKKX’16]
- First analysis as aPAKE and SaPAKE

# OPAQUE with HMQV [K'05]

**Init**: On input $pw, p_U$ by $U$ and $k, PS$ by $S$, $U$ computes $rw = H(pw, H'(pw)^k)$

and $c = AuthEnc_{rw}(p_U, P_U, P_S)$. $S$ stores $(k, p_S, c)$. $U$ only keeps $pw$.

**Login:**

$U$ $(pw)$ $\qquad\qquad$ $S$ $(k, p_S, c)$

$r, x \leftarrow Z_q$ $\qquad$ $\alpha = H'(pw)^r,\ X = g^x$ $\longrightarrow$

$y \leftarrow Z_q$

$\longleftarrow$ $\beta = \alpha^k,\ c,\ Y = g^y$

- $rw \leftarrow H(pw, \beta^{1/r})$
- $p_U, PK_U, PK_S \leftarrow AuthDec_{rw}(c)$
- $K = KE(p_U, x, P_S, Y)$ $\qquad\qquad$ $K = KE(p_S, y, P_U, X)$

HMQV:

For S: $\mathrm{KE}(p_s, x_s, P_u, X_u) = H\left((X_u P_u^{e_u})^{x_s + e_s p_s}\right)$

For U: $\mathrm{KE}(p_u, x_u, P_S, X_S) = H\left((X_s P_s^{e_s})^{x_u + e_u p_u}\right)$

# OPAQUE Performance (with HMQV)

- Single round (one message from client, one message from server)
  - OPRF and AKE can be done simultaneously
- Computational cost: comparable to the most efficient existing aPAKE

| | Per user | Per server | |
|---|---|---|---|
| SPAKE2+ [AP'05] | ~3.5 exps | ~3.5 exps | No rigorous proof |
| VTBPEKE [GW'17] | 4 exps | 4 exps | Not in UC |
| [JR'16] | 4 exps + 3 pairings | 4 exps + 3 pairings | Works in pairing groups only |
| OPAQUE | ~4.17 exps | ~3.17 exps | |

# OPAQUE Features

- Efficient and provable secure
  - Proof is modular: works for any UC OPRF and UC AKE-KCI
- Combination of good properties of aPAKE and password-over-TLS
  - Password only (non-PKI)
  - Server never sees password
  - Eliminates pre-computation attack (the only such protocol in non-PKI setting!)

# TLS Integration

- TLS Integration
  - Ciphertext c (sent from server to user) can contain user's other secrets, e.g. user's TLS signature key
  - Key exchange of OPAQUE can reuse that of TLS (no need to run two separate key exchanges): importance of generic composition
  - Protects user ID
- TLS protected by OPAQUE v. password protected by TLS

# OPAQUE Extensions

- Explicit authentication
  - Add one message (user sends $f_K(1)$, server sends $f_k(2)$ – server's message can be "piggybacked")
- Server-side threshold implementation
  - Use Threshold OPRF [JKKX'17]
  - Adversary needs to compromise a specific number ("threshold") of servers to launch offline dictionary attack

# OPAQUE: A Strong Asymmetric PAKE Protocol Secure Against Pre-Computation Attacks

# THANK YOU!

Stanislaw Jarecki, Hugo Krawczyk, **Jiayu Xu**

https://eprint.iacr.org/2018/163