Round-Optimal Password-Protected Secret Sharing and T-PAKE in the Password-Only Model

Stanislaw Jarecki Aggelos Kiayias Hugo Krawczyk

How to Protect a *Valuable Secret* When all You Remember is a *Password*

A motivating example: Bitcoin Wallet

- Stealing Bitcoin wallets is common news: How would you protect it?
  - smartphone? lose the phone, lose the wallet; add laptop? 2 stealing targets
- Backup in Internet server: *protection reduced to password*
  - online attacks (works for weak passwords)
  - offline server attacks: work even with reasonably secure passwords
- Obvious cryptographic solution: keep wallet encrypted in multiple locations; *secret share the encryption key* in multiple servers
  - But how do you authenticate to the servers? With a password, of course!
    - A strong independent password with each server? Not really
    - Same (or slight-variant) password for each server? Not good

➔ *Each server as a single point of failure!* Didn't achieve much, did we?

Password Protected Secret Sharing [BJSL'11]

- **Protection:** User secret shares a secret among n servers (threshold t); forgets the secret and keeps a single password.
- **Retrieval:** User contacts t + 1, or more, servers, authenticates using the single password and reconstructs the secret.
- Security guarantee: Attacker that breaks into t servers and finds all their secret information (including shares, long-term keys, password file, etc.) cannot learn anything about the secret (and password).
- Only adversary hope: Guess the password, try it in an online attack.
- Offline attacks with less than t+1 corrupted servers are useless.

\+ **Soundness:** User reconstructs the correct secret or else rejects.

PPSS: Security Definition

- As strong as possible: Only allows attacks that are *unavoidable*
- An attacker A can always test a guessed password p by one of:
    1. A interacts (as a user) with t+1 servers using password p; if A's execution accepts then guess was correct
        - It takes *online interactions* with t+1 servers to test a *single* password
    2. A simulates the sharing protocol with t+1 (imaginary) servers using password p and arbitrary secret s; then A interacts with U simulating the t+1 servers. If U accepts, the guess was correct.
        - Attacker controlling t+1 links to user can test a password
- Hence, if attacker controls t' servers and password chosen from D:

$$Adv_A \leq \left(q_U + \frac{q_S}{t-t'+1} \right) \cdot \frac{1}{|D|} + \varepsilon$$

More on our model

- Secure channels between user and servers assumed for initialization only (secret sharing phase)
- Reconstruction is in the CRS model (e.g., known EC group) – no PKI or secure channels assumed, not even between servers
  - ***user only remembers its password!***
  - Hedging property: If PKI available b/w user and servers, attack 2 is not possible (attacker advantage: $\frac{q_U}{|D|}+\varepsilon$)
- Robustness: If U can communicate without adversarial interference with t+1 servers, reconstruction succeeds (even if other links or participating servers are corrupted)

Comparison to Prior Work

- Bagherzandi-Jarecki-Saxena-Lu, CCS'11
  - Formalized PPSS notion as above (roots in t)
  - Scheme <u>assumes PKI</u> between user and servers, needs 3 (or 4) messages, 8t+7 exponentiations for client, 16 for each server

All 3 protocols in ROM. We also show a 4-msg std model.

- Camenish-Lehmann- Lysyanskaya-Neven, Crypto'14:
  - UC notion of PPSS (called PASS)
  - <u>no PKI</u> b/w client and servers (except at init) , auth'd channels b/w servers
  - 10 msgs, 14t+24 exponentiations for client, 7t+28 for each server
- Our scheme (follows BJSL definition)
  - **No PKI**, no authenticated channels (except for initialization)
  - **Single round (2 msgs), 2t+3 expon's for client, 2 for e/server**

From (t,n)-PPSS to (t,n)-threshold PAKE

- (t,n)-TPAKE: U can exchange keys securely w/ any subset of n servers using a single password as long as at most t servers are corrupted
  - exchange succeeds if undisturbed communication with t + 1 servers
- We prove a Generic composition theorem: PPSS + KE → T-PAKE.
- With the following property:

  Single-round PPSS → single round T-PAKE! (also w/PFS and PK KE)

→ First single-round T-PAKE:

  no prior work achieved that, not even assuming PKI and not even for special cases such as 2-out-of-2 (ours is also the most computationally efficient) A2

投影片 8

A2 holds even with forward secrecy (Diffie-Hellman) and with single-round public-key based KE (e.g. HMQV).
ADMINIBM, 2014/12/4

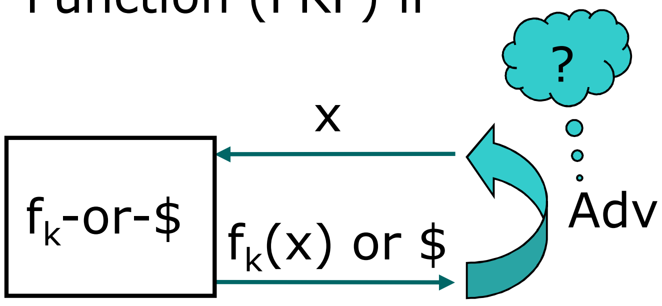
The PPSS Scheme

Highlights of Our PPSS Scheme

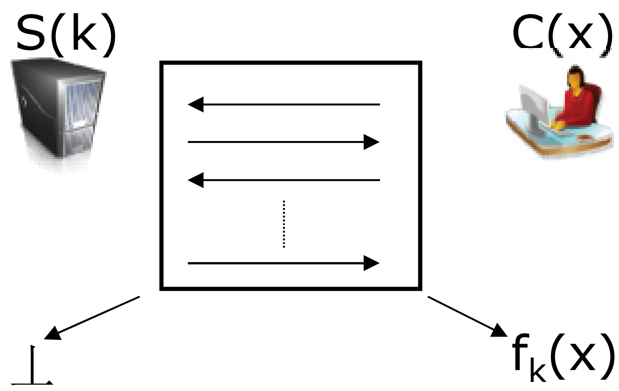
- One round (User to Server msg + Server to User msg)
- User performs 2 exponentiations per server
  - Undisturbed communication with t+1 servers suffices for reconstruction (and wrong secret never reconstructed)
- Server performs 2 exponentiations
- No inter-server communication
- No assumed PKI or secure channels (other than for initialization)

Main building block: Oblivious PRF (OPRF) [NR'04,FIPR'05]

$f_k(x)$ is a Pseudo-Random Function (PRF) if



OPRF protocol



- Fastest (2 exp's/party) is Hashed-DH PRF: $f_k(x) = H(H(x)^k)$,
- Oblivious computation via "Blind DH Computation":

C sends a = $[H(x)]^r$ to S, S replies with b = a^k, C sets $f_k(x) = H(b^{1/r})$,

Idea of Scheme (w/o validation steps)

- Initialization: User U on password p (server S_i has OPRF key k_i)
  - Chooses random s, secret shares s into $s_1, \dots, s_n$
  - Runs OPRF with server S_i, $1 \le i \le n$, to obtain $r_i = f_{ki}(p)$; encrypt s_i as $c_i = s_i \oplus r_i$
  - Stores c_i at S_i ; erases all info; memorizes p.
- Reconstruction: User U on password p
  - Receives c_i from S_i and runs OPRF to recover $r_i = f_{ki}(p)$; sets $s_i = c_i \oplus r_i$
  - Reconstructs s from (subset of) $s_1, \dots, s_n$
- For soundness: At initialization, U sets K||r= PRG(s), stores C=Commit(pw; r) at each server S_i. K is defined as the secret key for reconstruction.

  At reconstr'n, U gets C from S_i, sets K||r= PRG(s); checks C = Commit(pw; r). If check succeeds U outputs K, else it rejects (can use any C that t+1 agree with)

Adding Validation

- Actual protocol uses "verifiable OPRF" where user can verify correct computation of $f_k(p)$.
- For this, we assume S_i commits to its function f_{ki} via a descriptor π_i
- The commitment Commit(p; r) is augmented to Commit(p, **c**, $\boldsymbol{\pi}$; r) with **c** = $(c_1,\dots,c_n)$, $\boldsymbol{\pi} = (\pi_1,\dots,\pi_n)$, and values **c** and $\boldsymbol{\pi}$ are stored at each S_i
- U can try reconstruction on any subset of t+1 servers that agree on the values C, **c** and $\boldsymbol{\pi}$. User accepts if commitment verifies correctly.

- For the DH-OPRF solution $f_{k_i}(x) = H(H(x)^{k_i})$, we set $\pi_i = g^{k_i}$ and add to the protocol a DDH NIZK.
  - In progress: Relax verifiability, get rid of NIZK (except for robustness)

PPSS Protocol (for DH OPRF)

- **Init:** Server S_i has key k_i to OPRF $f_{k_i}(x) = H(H(x)^{k_i})$, denote $\pi_i = g^{k_i}$
  User U (on password p and servers' functions $\pi_1, \ldots, \pi_n$)
  - Chooses random s, secret share s into $s_1, \ldots, s_n$.
  - Runs OPRF with server S_i to obtain $r_i = f_{ki}(p)$; sets $c_i = s_i \oplus r_i$.
  - Defines $\mathbf{c} = (c_1,\ldots,c_n)$, $\boldsymbol{\pi} = (\pi_1,\ldots,\pi_n)$, and Com = Commit(p, $\mathbf{c}$, $\boldsymbol{\pi}$; r) where $K||r \leftarrow PRG(s)$; Stores at each server S_i: w = ($\mathbf{c}$, $\boldsymbol{\pi}$, Com).
  - K is defined as the recoverable key
- **Reconstruction:** For each S_i: receive w_i from S_i; set w to majority w_i; run OPRF to get $r_i = f_{ki}(p)$ (verify using π_i from w); set $s_i = c_i \oplus r_i$.
- Reconstruct s from s_i's ; set $K||r \leftarrow PRG(s)$; set C=Commit(p, $\mathbf{c}$, $\boldsymbol{\pi}$; r); reject if C differs from Com value in w, otherwise output K.

Defining "Verifiable OPRF"

- OPRF notion is intuitive: Secure two-party computation of $f_k(x)$ where one party holds k and one holds x
- Yet, defining OPRF security is challenging:
  - E.g.: Secure 2-PC may impose input extraction, prevents concurrency, requires secure channels (all elements we want to avoid)
  - Indistinguishabilty definition tricky too: What's the test for the attacker after running q protocol executions (on unknown inputs)?
- We formulate a UC definition of "Verifiable OPRF" (user can check that the server uses same function consistently: e.g., always same output on pwd)
  - We bypass input extraction via ticketing mechanism
    - per-server ticket: increases with each server call, decreases with server output, no output from functionality if ticket = 0
- We show instantiations in ROM (DH, RSA), under one-more assumpt'n, and standard model (NR)

Comparison to Prior Work (PPSS and T-PAKE)

Achieving single-round password-only protocol in the CRS and ROM models for arbitrary (n, t) parameters with no PKI requirements for any party and no inter-server communication (except for server authentication at initialization).

scheme	$(t+1, n)$	ROM/std	client	inter-server	msgs	total comm.	comp. C | S
BJKS [2]	$(2,2)$	ROM	PKI	PKI	7	O(1)	O(1)
KMTG [6]	$(2,2)$	Std/ROM	CRS	sec.chan.	≥ 5	O(1)	O(1)
CLN [4]	$(2,2)$	Std/ROM	CRS	PKI	8	O(1)	O(1)
DRG [5]	$t<n/3$	Std	CRS	sec.chan.	≥ 12	$O(n^3)$	$O(1)$ | $O(n^2)$
MSJ [7]	any	ROM	PKI	PKI	7	$O(n^2)$	$O(1)$ | $O(n)$
BJSL [1]	any	ROM	PKI	PKI	3	$O(t)$	$8t+17$ | 16
CLLN [3]	any	ROM	CRS	PKI	10	$O(t^2)$	$14t+24$ | $7t+28$
Our PPSS1	any	ROM	CRS	none	2	$O(t \log n)$	$2t+3$ | 2
Our PPSS2	any	Std	CRS	none	4	$O(\ell t \log n)$	$O(t\ell)$ | $O(l)$

Thanks!

http://eprint.iacr.org/2014/650