Server-Assisted Generation of a Strong Secret from a Password

Warwick Ford, VeriSign, Inc.
(Joint research with Burt Kaliski, RSA Laboratories)
Requirement

- User who roams between client terminals needs to
  - obtain private key or data
  - strongly authenticate to application servers
- No local stored state
- No smartcards
- Private data downloaded from online credentials server
Traditional Credentials Server Solution

- Surveyed in Perlman & Kaufman, NDSS ‘99
  - Examples EKE, SPEKE
- Protocol exposes no information about private data
- Throttling/lockout:
  - Limits password guessing
  - Makes friendly passwords possible
  - Based on failed password authentications
Weakness in Traditional Design

- If server compromised, attacker can potentially:
  - Attack credentials database, e.g., password verifiers by exhaustive attack (even if passwords not determinable directly)
  - Disable throttling/lockout and exhaustively attack with password guesses
- Vulnerable to password attack
- Password exposure means private data exposure
- Many users may be compromised in one attack
Objective: Compromise of one server exposes neither private data nor password

Not as easy as it looks

- Ordinary secret-sharing not adequate if servers have to verify passwords
Basic Approach

- Client generates strong master secret $K$ via interaction with two or more servers
- Client proves successful regeneration of $K$ to all servers
- $K$ can unlock encrypted private data or facilitate authentication to other servers
- No server can learn $K$ or password
In More Detail…

- **Pre-knowledge**
  - User knows password $P$
  - Each server $S_i$ holds its own secret $d_i$ for that user
  - Each $S_i$ also holds its own strong verifier $K_i$ for $K$

- **Client generates strong master secret $K$**
  - For each $S_i$, client computes strong secret $R_i$
    - via a password hardening transaction depending on $P$ and $d_i$
    - subject to throttling/lockout
  - Combines all the $R_i$ to give $K$

- **Client proves successful regeneration of $K$ to servers**
  - For each server $S_i$ generates strong verifier $K_i$ from $K$
  - Demonstrates knowledge of $K_i$ to server $S_i$

- $K$ can unlock encrypted private data or facilitate authentication to other servers
Secret-Strengthening Protocol

Properties:

- $R_1$ is a strong secret
- Observer cannot feasibly learn $R_1, d_1$ or $P$
- Server cannot feasibly learn $R_1$ [or $P$ ?]
- Same $R_1$ always generated for same $P$

User U

Password $P$ entered

Generate random $k$

$w = f(P)$

$r = w^k \mod p$

$R_1 = s_1^{1/k} \mod p = w^{d_1} \mod p$

Server $S_1$

$s_1 = r^{d_1} \mod p$

$U, r$

$U, d_1$
Do It with Two Servers

Properties:

- \( K \) is a strong secret
- Observer cannot feasibly learn \( K \) or \( P \)
- Neither server can feasibly learn \( K \) or \( P \)
- Same \( K \) always generated for same \( P \)
- Both servers need to cooperate for \( K \) to be generated
Now Prove It was Successful

Properties:

- Each server gets proof that client knows $K$
- Server’s knowledge of $K_i$ does not feasibly assist determining $K$ (or password)
Some Variants

- Other secret-strengthening protocols
  - ECC variant is obvious
  - RSA-based also exists

- Other verification methods
  - $K$ decrypts a private digital signature key; signed nonce proves regeneration to server holding public key

- Use threshold functions in combining hardened passwords

- Use other functions of master secret to authenticate to other (application) servers
A Special Case Variant

- Client interacts with password hardening server $S_1$ to obtain $R_1$
- Client uses $T_1$ derived from $R_1$ to authenticate to a second server $S_2$
- $S_2$ confidentially delivers to client: secret $K$ encrypted under $T_2$ derived from $R_1$
- Client decrypts $K$
- Client verifies to $S_1$ by proving regeneration of $K$
Special Case Variant - Protocol

User U

Password $P$ entered

$w = f(P)$

Generate random $k$

$r = w^k \mod p$

$R_1 = s_1^{1/k} \mod p$

$T_1 = OWF(R_1, 1)$

$T_2 = OWF(R_1, 2)$

$K = D_{T_2}(E_{T_2}(K))$

Server $S_1$

$s_1 = r^{d_1} \mod p$

$U, d_1$

Secure channel

Then prove knowledge of $K$ to $S_1$

Properties:

- Attractive when $S_2$ already exists (e.g., SSL or SPEKE server)
- Adding one password hardening server $S_1$ provides the requisite added strength

Secure channel
The Fundamental Characteristics

- Must recover a master secret using more than one independent server
  - all of which contribute to recovering the secret
  - all of which employ throttling/lockout
- At least one secret-contributing server must use secret-strengthening
- Must prove successful regeneration of a strong secret to at least two verification servers
Non-Repudiation Ramifications

- Single server design is weak wrt non-repudiation
  - User can plausibly claim that insider/penetrator at the server recovered the private key and signed

- The multi-server design significantly improves non-repudiation
  - It is much harder to mount a plausible argument that independently controlled servers colluded

- But, claims of non-repudiability still rest on confidence that the client terminal is secure
  - There is no silver bullet for this concern
Summary of the Technology

- Traditional credentials server architecture is vulnerable to server compromise and exhaustive password guessing against stored password-derived values
  - Server vulnerability raises security concerns and kills non-repudiation
- Need multiple independent servers contributing to secret regeneration
  - Each must independently throttle/lockout
- Need password hardening as a basis of establishing strong secret from weak secret
Current-shipping VeriSign enterprise PKI offering includes the option:

- Two-server secret-strengthening technology to support protection of private key plus arbitrary user data
- Servers may be operated by Enterprise and/or VeriSign

Alternative packagings (e.g., for SSO, Aggregation) in development
For More Information

➢ See Ford/Kaliski WETICE 2000 paper at:

➢ Contact details:
  Warwick Ford, VeriSign, Inc.
  E-mail: wford@verisign.com
  Tel: (781) 245 6996 x225