基于移动端协助的远程用户单一口令认证方法

徐渊[1]，杨超[2]，杨力[3]

（1. 西安财经大学实验实训教学管理中心，陕西 西安 710100；2.西安电子科技大学信息网络技术中心，陕西 西安 710071；
3.西安电子科技大学网络与信息安全学院，陕西 西安 710071）

摘 要：针对口令认证系统中用户频繁重复使用同一弱口令的问题，提出一种基于服务器与便携移动设备间秘密共享的单一口令认证方法，允许远程用户使用单一口令和多个在线服务进行安全认证，且客户端 PC 无需存储用户的任何秘密信息；即使移动设备丢失或被盗，也不会损害用户信息。安全性分析与性能测试结果表明，新方法大大提高了用户私密信息的安全性，可以抵御字典攻击、蜜罐攻击、跨站点编程攻击及网络钓鱼攻击，减轻用户记忆负担，缓解存储压力，易于部署。

关键词：口令认证；秘密共享；移动端辅助认证；恶意软件；字典攻击

中图分类号：TP391

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2019044

Single password authentication method for remote user based on mobile terminal assistance

XU Yuan[1], YANG Chao[2], YANG Li[3]

1. Experimental Teaching Management Training Center, Xi’an University of Finance and Economics, Xi’an 710100, China
2.Information Network Technology Center, Xidian University, Xi’an 710071, China
3. School of Cyber Engineering, Xidian University, Xi’an 710071, China

Abstract: To address the issue that users frequently reuse their weak passwords in password-based authentication system, single password authentication based on secret sharing between server and mobile terminal (SPASS) was proposed, which allows a remote user to use a single password to authenticate to multiple services securely and has no need to store any secret of the user in the client PC. Even when the mobile device is lost or stolen, no damage to the user’s information will be induced. Security analysis and performance test show that SPASS greatly improves the security of the user’s secret information and resists dictionary attacks, honeypot attacks, cross-site scripting attacks etc. Furthermore, the proposed scheme can lighten burden of the user’s memory, reduce the storage pressure and easy to be deployed.

Key words：password-based authentication, secret sharing, authentication based on mobile terminal, malware, dictionary attack

1 引言

随着互联网的迅速发展和应用系统的不断增加，使用口令认证机制来判断远程用户的身份已经变得最为常见。用户仅需使用其预先设置的口令与各种服务器进行认证，认证成功后便可获得应用服务。

然而，用户往往会设置简单易记的口令，且频繁使用。研究发现[1]一般用户会在 3 个月内登录 25 个不同的在线服务，但却只有 7 个口令，若其中之

收稿日期：2018–07–20；**修回日期**：2018–09–23

基金项目：国家重点研发计划基金资助项目（No.2017YFGX110123）；陕西省科技创新计划基金资助项目（No.201809168CX9JC10）；国家自然科学基金资助项目（No.61672415）；西安财经大学 2018 年度教育教学改革研究基金资助项目（No.18xcj36）

Foundation Items: The National Basic Research Program of China (No.2017YFGX110123), The Science and Technology Innovation Planning Project of Shaanxi Province (No.201809168CX9JC10), The National Natural Science Foundation of China (No.61672415), The Research Program of Education and Teaching Reform of Xi’an University of Finance and Economics in 2018 (No.18xcj36)

一泄露，将会威胁其他账户信息的安全。当用户忘记某个欲登陆的在线服务的口令时，会逐一尝试自己的口令，直到认证成功。如果该服务器是恶意的，它除了获得此服务的口令外，还会得到用户的其他口令，可能会对用户的其他账户发起跨站点的伪装攻击。即使服务器是所谓的可信服务器，也可能会对用户发起同样的攻击。Facebook 的 CEO 曾在 2004 年使用 Facebook 的登录数据访问一些商业对手和记者的私人邮件[2]。

1992 年，Bellovin 和 Merritt 在 Diffie-Hellman 密钥交换协议的基础上，提出了口令认证密钥交换（PAKE, password-authenticated key exchange）协议的最早例子——加密密钥交换协议（EKE, encrypted key exchange）[3]，此协议实现了双向认证，可以在用户和服务器之间建立一条安全的认证信道。因此，专家们都在此基础上进行研究，衍生出许多变化版本[4-5]，但它们都无法抵御恶意服务器和跨站点伪装。APAKE（asymmetric password-authenticated key exchange）协议要求只有用户知道口令，服务器存储口令的一个单向函数[6-7]，然而，该协议还是容易受到服务器的字典攻击以及电脑黑客从服务器窃取信息的影响。Boyen[8]表明任何基于口令的认证协议（协议只涉及客户和服务器双方）都会受到服务器的字典攻击。服务器总是会尝试每一个可能的单一口令，看其是否可以认证成功。Boyen 的 HPAKE（hardened password authenticated key exchange）协议[9]让用户控制字典攻击的代价；用户需选择一个安全参数 τ，在注册和认证阶段执行 $\theta(\tau)$。由此可见，早期的口令认证机制的一个内在限制就是认证协议只涉及两方，即客户端和服务器端。

解决双方口令认证机制内在限制的一个有效方法就是增加一个信任的移动设备且可由用户自己携带，这样的设备（如智能卡）[10-12]，可以完全消除对口令认证的需求。但是，文献[10]不能很好地衡量多个在线服务；文献[11]只将离线字典猜测攻击作为目标，却忽视了内部攻击和拒绝服务攻击的危害；而文献[12]无法完全抵抗口令猜测攻击，且需要大量的计算开销。基于此，文献[13-14]提出了结合口令、智能卡和生物特征的三因子认证方案，该方案虽然具有一定的优势，但还是无法解决智能卡带来的不安全和不方便问题。文献[15]做出了很大的改进，但是方案的局限性在于需要依赖于服务器的公钥来完成认证。

早在 2004 年，文献[16]提出由手机作为便携的移动设备。但是在该方案中，还需要增加一个安全可信的代理方，这使得系统部署较为繁琐；而且手机端和代理方需要进行多次交互，更容易遭受中间人攻击，因此，该方案无法广泛地应用于实际认证中。方案[17]在此基础上，提出了简易有效的基于指纹与移动端协助的口令认证方法，但是该方法会受限于手机对指纹的提取技术。

文献[18-19]提出的方案，都是利用手机扫描二维码的形式对认证时的秘密信息进行加密或解密操作。但文献[18]的方案中，用户需要在设备上添加事先预设的条形码，当进行多个在线服务时，预设步骤较为繁琐，而且所需的存储量较大，因此该方案可操作性不强；文献[19]的方案中，每个二维码为某个对象或应用提供了一个唯一标识符来自动检测系统、贸易、工业、医院等，任何一个带有视频采集功能的读写器都可以直接从标签中读出内容。当条形码包含私密信息时，就会造成数据私密信息的安全风险。

SPA（single password authentication）[20]方案要求额外增加一个存储设备（云存储器或可信的手机设备）来存储用户的数据，以此来辅助用户进行注册和认证。当存储设备是云存储器时，利用 HCR (hidden credential retrieval from a reusable password)[8]协议的思想，在不可信的云存储器上存储认证密文，但其需要设置安全信道，这在实际应用中很难实现。此外，其还需要假设云存储器和在线服务不能“勾结”，否则会带来严重的安全问题。当存储设备是可信的手机设备时，强制要求此手机一直是完全可信的，不存在任何恶意代码侵袭，而且也不能与服务器进行“勾结”，若手机丢失或被偷，则用户存储在手机上的密文信息就很有可能被对手破解。这一系列的缺陷大大降低了协议的可用性和安全性。

PPSS（password-protected secret sharing）[21]是 Bagherzandi、Jarecki 和 Saxena 在 2011 年的 CCS（computer and communications security）会议上提出的，允许用户在多个服务器之间来分配个人数据，以便使用单一且用户易记的口令来恢复该个人数据，其中任何一个单独的服务器，甚至某些服务器相互“勾结”（勾结数量必须小于一定的规模），也无法对口令嵌入离线字典攻击或获取到任何关于个人数据的信息。接着，Camenisch、Lysyanskaya 和 Neven

对 PPSS 进行改进，在 2012 年首次提出 2PASS（two-server password-protected secret sharing）[22]，实现了在 2 个服务器之间来分配个人数据。但是，这些协议都是用来在不可信的服务器上存储用户个人数据信息，并没有将其用于在线服务的认证协议中。此外，它们都需要额外服务器（至少 2 个）的协助，给用户的日常使用和管理造成了很大的不便。

本文提出了一种基于服务器与移动设备间秘密共享的单一口令认证方法（SPASS, single password authentication based on secret sharing between server and mobile device），该方法不需要增加额外的服务器，减少了服务器之间相互识别所耗费的时间和存储空间，实现了在有便携式移动设备参与的情况下，仅使用单一口令就可以和多个在线服务进行安全认证，进一步提高了用户私密信息的安全性。

2 系统模型与敌手模型

2.1 系统模型

SPASS 方案中的移动设备包含平板电脑、智能手机等，本文以智能手机作为移动设备为例进行详细讨论。系统架构模型如图 1 所示，主要包含个人电脑（PC，personal computer）、云服务器（CS，cloud server）和移动智能手机（MP，mobile smart phone）三部分。图 1 中①和②属于注册阶段，③、④和⑤属于认证阶段。

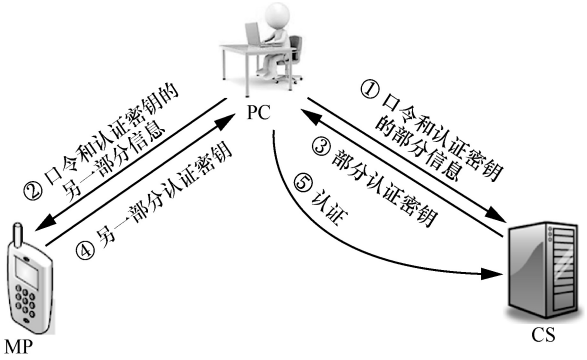

图 1 系统模型

云服务器（CS）：负责在注册阶段存储用户口令、认证密钥的部分信息；认证阶段验证用户口令，若正确，向客户端提供其所存储的认证密钥的部分信息，最后完成认证。要求 CS 能够接入互联网，并具有必要的存储空间。

个人电脑（PC）：主要负责在注册阶段录入用户注册信息且生成随机的认证密钥；认证阶段对检索到的密钥参数进行运算处理。要求 PC 可以连接到互联网，能够远程登录 CS 并与其进行通信。

移动智能手机（MP）：注册阶段存储用户口令、认证密钥的另一部分信息；认证阶段协助 CS 完成对口令的验证后，为用户提供其所存储的认证密钥的另一部分信息。MP 除了具有基本的功能外，还能够通过 Wi-Fi 功能连接到 CS，以便与 CS 进行通信，并且也需要一定的存储空间。

2.2 敌手模型

上述系统模型主要包括 3 种实体，即云服务器、个人电脑和移动智能手机，仅当实体可信时，系统才会诚实执行此协议。因此，其对应的敌手模型可总结如下。

1) 云服务器是半可信的。假设注册阶段的云服务器是可信的，则用户已经成功完成个人信息注册。此后，当对手发起主动攻击时，云服务器就有可能遭受来自外部对手的重放攻击或字典攻击，也可能会遭受系统内部攻击，即其自身可能是恶意的，只在线下对已存储的用户密文信息发起攻击。

2) 个人电脑是不可信的。当用户发起认证请求时，其需要在客户端 PC 输入用户名和口令，而此时 PC 也许已经遭受到恶意软件或病毒的侵袭，敌手会发起被动攻击，利用键盘记录器记录下用户的口令，从而进一步地获取用户私密信息。

3) 移动智能手机是不可信的。用户在注册阶段和认证阶段都会用到移动智能手机，而手机可能丢失、被偷或者被恶意软件侵害，则对手可能获得存储在手机上的部分用户认证信息，从而试图恢复原始信息并发起主动攻击，冒充用户与服务器进行通信。

3 SPASS 方案设计实现

3.1 SPASS 方案设计思想

本方案将认证密钥和口令编码后产生的随机串分配在 CS 和 MP，进行认证时，用户需要从 CS 和 MP 检索到这些随机串，然后恢复出完整的口令和认证密钥。这样，用户不仅可以与不可信的 MP 进行交互，而且不需要在 PC 存储任何个人隐私数据，不论 MP 或 CS 任何一方遭受攻击，都不会影响信息的安全。此外，本方案中用户 PC 与 CS 的交互、MP 与 CS 的交互都不需要假设安全信道，也不需要添加额外的服务器。

3.2 SPASS 方案详细设计

假设如下。1) 公共参考字符串的泛函数 F_{CRS}

描述由 GGen(1^k)生成的一个阶为素数 q 的群 G 和生成元 g，以及由(keyg, enc, dec)产生的公钥 PK，且其对应的私钥是未知的。2) 系统中，已被 F_{CA} 证实的所有服务器的公钥是存在的，而且不要求用户拥有这样的公钥。更确切地说，假设参与认证的服务器已经由(keyg_2，enc_2，dec_2)生成了密钥对(PE，SE)、由(keysig，sig，ver)生成了密钥对(PS_1，SS_1)，而智能手机只需要通过(keysig, sig, ver)生成密钥对(PS_2，SS_2)。此外，已经通过使用(Register, S, (PE，PS_1))来调用 F_{CA} 实现了该服务器公钥的注册，使用(register，M，PS_2)来调用 F_{CA} 实现了该智能手机公钥的注册。

此外，方案要求参与认证的 CS 与 MP 在认证阶段需要向对方证明其所陈述信息的正确性（本质上，加密是正确的计算）。因此，在方案的描述中，定义 ZK{(w) : predicate(w,y)= 1}来表示可以证明某个关于公共值 y 和证据值 w 的描述是正确的。

由于本方案中没有假设安全信道，消息可能来自任何一方，因此参与方之间发送给彼此的所有消息都应该加上标签(reg，sid，qid)或(aut，sid，qid)，以及与各自的具体步骤相一致的序号。

注册阶段和认证阶段的具体步骤分别描述如下。

1) 注册阶段

在这个阶段，用户需要首先输入(reg，sid，p，K)，其中 sid=(name，CS，MP)，name 是用户选择的用户名；p 是用户选择的口令，$K \to \text{MACKeyGen}(1^k)$ 是随机产生的认证密钥，假设 p 和 K 都已经被编码为 G 中的元素；MP 和 CS 的内部永久性存储分别表示为 st_{MP} 和 st_{CS}。注册阶段的具体协议如图 2 所示。

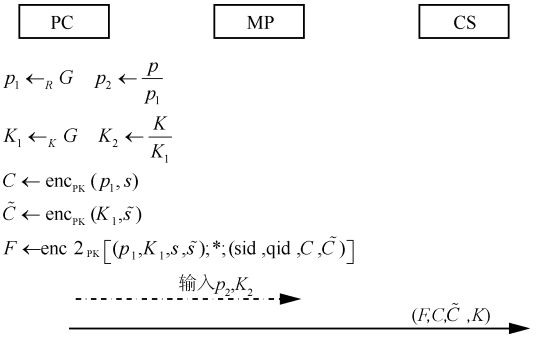

图 2　注册阶段的具体协议

步骤 1　用户 PC 执行如下计算。

① 询问 F_{CRS} 获得 PK，发送(authentication，sid，CS)给 F_{CA} 获得(PE，PS_1)。

② 选择 $p_1 \leftarrow_R G$ 和 $K_1 \leftarrow_R G$，计算 $p_2 \leftarrow \frac{p}{p_1}$ 和 $K_2 \leftarrow \frac{K}{K_1}$。

③ 选择随机数 $s, \tilde{s} \leftarrow_R Z_q$，在 CRS 下加密 p_1 和 K_1，即 $C \leftarrow \text{enc}_{\text{PK}}(p_1; s)$，$\tilde{C} \leftarrow \text{enc}_{\text{PK}}(K_1; \tilde{s})$。

在服务器的公钥下加密，即 $F \leftarrow \text{enc}2_{\text{PE}}((p_1, K_1, s, \tilde{s}); *; (\text{sid}, \text{qid}, C, \tilde{C}))$。

④ 发送(F, C, $\tilde{C}$, K)给 CS。

步骤 2　CS 执行如下操作。

① 收到来自用户的消息，解析为(Reg, sid, qid, 1, F, C, $\tilde{C}$, K)。

② 发送(authentication，sid，MP)给 F_{CA}，获得 PS_2。

③ 用标签(sid, qid, C, $\tilde{C}$)解密 F，检查是否满足 $C \leftarrow \text{enc}_{\text{PK}}(p_1; s)$，$\tilde{C} \leftarrow \text{enc}_{\text{PK}}(K_1; \tilde{s})$。

④ 检查是否在状态中没有记录的 $\text{st}_{\text{CS}}(\text{sid})$。

⑤ 计算签名 $\sigma \leftarrow \text{sig}_{\text{SS}_1}(\text{sid}, \text{qid}, C, \tilde{C})$，并且将 σ 发送给 MP。

步骤 3　MP 执行如下操作。

① 解析来自 CS 的消息为(reg，sid，qid，1，σ)。

② 发送(authentication，sid，CS)给 F_{CA}，获得(PE，PS_1)。

③ 检查是否 $\text{ver}_{\text{PS}_1}((\text{sid}, \text{qid}, C, \tilde{C}), \sigma) = 1$。

④ 检查是否在状态中没有记录的 $\text{st}_{\text{MP}}(\text{sid})$；然后直接输入 p_2 和 K_2。

⑤ 计算签名 $\tau_2 \leftarrow \text{sig}_{\text{SS}_2}(\text{sid}, \text{qid}, C, \tilde{C}, \text{succ})$，发送给 CS。

⑥ 更新状态 $\text{st}_{\text{MP}}[\text{sid}] \leftarrow (\text{PS}_1, p_2, K_2, C, \tilde{C})$，输出(Reg，sid，qid，succ)。

步骤 4　CS 执行如下操作。

① 解析来自 MP 的消息，检查是否 $\text{ver}_{\text{PS}_2}((\text{sid}, \text{qid}, C, \tilde{C}, \text{succ}), \tau_2) = 1$。

② 计算 $\tau_1 \leftarrow \text{sig}_{\text{SS}_1}(\text{sid}, \text{qid}, C, \tilde{C}, \text{succ})$，并且发送给用户 PC 端。

③ 更新状态 $\text{st}_{\text{CS}}[\text{sid}](\text{PS}_2, p_1, K_1, s, \tilde{s}, C, \tilde{C}, K)$，输出(Reg，sid，qid，succ)。

步骤 5　用户 PC 执行如下操作。

解析来自 CS 的消息，检查是否 $\mathrm{ver}_{\mathrm{PS}_1}((\mathrm{sid},\mathrm{qid},C,\tilde{C},\mathrm{succ}),\tau_1)=1$。如果是，输出(reg，sid，qid，succ)。

2) 认证阶段

此时，用户输入(aut, sid, qid', p')，MP 和 CS 将它们各自的状态信息 $\mathrm{st}_{\mathrm{MP}}$ 和 $\mathrm{st}_{\mathrm{CS}}(\mathrm{sid})$作为输入。其中考虑安全因素，用户此时输入的口令可能正确，也可能不正确，因此，将其记为 p'。认证阶段的具体协议如图 3 所示。

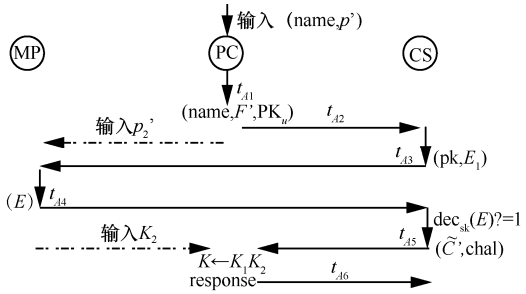

图 3 认证阶段的具体协议

步骤 1 用户 PC 执行如下计算。

① 询问 F_{CRS} 获得 PK，发送(authentication, sid, CS)给 F_{CA} 获得(PE, PS_1)。

② 选择 $p_1' \leftarrow {}_R G$，计算 $p_2' \leftarrow \frac{p'}{p_1'}$；产生$(\mathrm{PK}_u, \mathrm{SK}_u) \leftarrow \mathrm{keyg}(1^k)$。

③ 选择随机数 $s' \leftarrow {}_R Z_q$，在 CRS 下加密 p_1'，即 $C' \leftarrow \mathrm{enc}_{\mathrm{PK}}(p_1';s')$；在服务器的公钥下加密，即 $F' \leftarrow \mathrm{enc2}_{\mathrm{PE}}((p_1';s');*;(\mathrm{sid},\mathrm{qid}',C',\mathrm{PK}_u))$。

④ 发送密文(F'，C'，PK_u) 给 CS。

步骤 2 CS 执行如下操作。

① 解析来自用户的消息为(aut，sid，qid'，1，F'，C'，$\mathrm{PK_u}$)。如果不存在状态信息 $\mathrm{st}_{\mathrm{CS}}(\mathrm{sid})$，则失败；否则恢复 $\mathrm{st}_{\mathrm{CS}}[\mathrm{sid}] \leftarrow (\mathrm{PS}_2, p_1, K_1, s, \tilde{s}, C, \tilde{C}, K)$。

② 向外界输出 $(\mathrm{ANot},\mathrm{sid},\mathrm{qid}')$，等待输入 $(\mathrm{Perm},\mathrm{sid},\mathrm{qid}',a)$，其中 $a \in \{\mathrm{allow},\mathrm{deny}\}$，如果 $a=\mathrm{deny}$，则失败。

③ 用标签(sid, qid', C', PK_u)解密 F，如果标签错误，则失败。

④ 检查是否 $C' = \mathrm{enc}_{\mathrm{PK}}(p_1';s')$。

⑤ 产生适合于同态加密方案的密钥对 (pk,sk) $\mathrm{keyg}(1^k)$；选择 $r_1 \leftarrow {}_R Z_q$，计算 $E_1 \leftarrow \mathrm{enc}_{\mathrm{pk}}(\frac{p_1}{p_1'};r_1)$。

⑥ 计算签名 $\sigma_1' \leftarrow \mathrm{sig}_{\mathrm{SS}_1}(\mathrm{sid},\mathrm{qid}',C',\mathrm{PK}_u,E_1,\mathrm{pk})$，将 $(\mathrm{pk},E_1,\sigma_1')$ 发送给 MP。

⑦ 向 MP 证明 E_1 是正确的。$\pi_1 := \mathrm{ZK}\{(p_1, p_1', s, s', r_1): E_1 = \mathrm{enc}_{\mathrm{pk}}(\frac{p_1}{p_1'};r_1) \wedge C = \mathrm{enc}_{\mathrm{PK}}(p_1;s) \wedge C' = \mathrm{enc}_{\mathrm{PK}}(p_1';s')\}$。

步骤 3 MP 执行如下操作。

① 输入 $(\mathrm{Aut},\mathrm{sid},\mathrm{qid}',p_2')$，如果不存在状态信息 $\mathrm{st}_{\mathrm{MP}}(\mathrm{sid})$，则失败；否则恢复 $\mathrm{st}_{\mathrm{MP}}[\mathrm{sid}] \leftarrow (\mathrm{PS}_1, p_2, K_2, C, \tilde{C})$。

② 向外界输出 $(\mathrm{ANot},\mathrm{sid},\mathrm{qid}')$，等待输入 $(\mathrm{Perm},\mathrm{sid},\mathrm{qid}',a)$，其中 $a \in \{\mathrm{allow},\mathrm{deny}\}$。如果 $a=\mathrm{deny}$，则失败。

③ 解析来自 CS 的消息为 $(\mathrm{Aut},\mathrm{sid},\mathrm{qid}',\mathrm{l},\mathrm{pk},E_1,\sigma_1')$；并且与 CS 相互作用来检查证据 π_1。验证签名是否满足 $\mathrm{ver}_{\mathrm{PS}_1}((\mathrm{sid},\mathrm{qid}',C',\mathrm{PK_u},E_1,\mathrm{pk}),\sigma_1')=1$。

④ 选择随机数 r_2，$z \leftarrow {}_R Z_q$，计算 $E_2 \leftarrow \mathrm{enc}_{\mathrm{pk}}(\frac{p_2}{p_2'};r_2)$，$E \leftarrow (E_1 \times E_2)^Z$。

⑤ 计算 $\sigma_2' \leftarrow \mathrm{sig}_{\mathrm{SS}_2}(\mathrm{sid},\mathrm{qid}',C',\mathrm{PK_u},E)$，将 (E,σ_2') 发送给 CS。

⑥ 向CS证明E是正确的。$\pi_2 := \mathrm{ZK}\{(p_2, p_2', r_2, z): E = (E_1 \mathrm{enc}_{\mathrm{pk}}(p_2 / p_2';r_2))^Z\}$。

步骤 4 CS 执行如下操作。

① 将来自 MP 的消息解析为 (E,σ_2')，且和 MP 相互作用核实证据 π_2。

② 验证签名是否满足 $\mathrm{ver}_{\mathrm{PS}_2}((\mathrm{sid},\mathrm{qid}',C',\mathrm{PK}_u,E),\sigma_2')=1$ 以及 $z\neq 0$，即 $E \neq \mathrm{enc}_{\mathrm{pk}}(1;0)$。

③ 用 sk 解密 E，看结果是否为 1。

④ 向 MP 证明 E 解密后结果为 1。$\pi_3 := \mathrm{ZK}\{(\mathrm{sk}): 1 = \mathrm{dec}_{\mathrm{sk}}(E)\}$。

⑤ 选择随机数 $\tilde{s}' \leftarrow {}_R Z_q$，计算密文 $\tilde{C}' \leftarrow \mathrm{enc}_{\mathrm{PK}_u}(K_1,\tilde{s}')$ 和签名 $\tau' \leftarrow \mathrm{sig}_{\mathrm{SS}_1}(\mathrm{sid},\mathrm{qid}',C',\mathrm{PK}_u,\tilde{C}')$。在其数据库中查阅与 sid 相联系的密钥 K，选择随机挑战 $\mathrm{chal} \leftarrow (0,1)^k$，发送消息 $(\tilde{C}',\tau',\mathrm{chal})$ 给用户 PC 端，并且输出 $(\mathrm{Aut},\mathrm{sid},\mathrm{qid}',\mathrm{succ})$。

步骤 5 MP 执行如下操作。

① 与 CS 一起核实证据 π_3。

② 直接在 PC 端输入 MP 存储的 K_2。

步骤 6 用户 PC 执行如下操作。

① 解析来自 CS 的消息，验证签名 $\mathrm{ver}_{\mathrm{PS}_1}((\mathrm{sid},$

$\mathrm{qid}^{'},C^{'},\mathrm{PK}_u,\tilde{C}^{'});\tau^{'})=1$，解密密文 $\tilde{C}^{'}$，得到 $K_2\leftarrow$，$\mathrm{dec}_{\mathrm{SK}_u}(\tilde{C}^{'})$ 恢复认证密钥 $K\leftarrow K_1K_2$。

② 用认证密钥 K 和随机挑战 chal 进行计算，得到认证响应 response $\leftarrow$ MAC(K,chal)。向 CS 发送认证响应 response 进行认证。

步骤 7　CS 继续执行如下操作。

计算 MAC(K,chal) 并与接收到的 response 进行对比，一致则接受此次登录认证，否则拒绝登录认证。

4　SPASS 方案的安全性证明与分析

4.1　SPASS 方案安全算法的模块划分

SPASS 方案包含以下 6 个算法模块。

1) userGen 模块：该算法用于生成用户名 name 和口令 pwd。

2) register 模块：用户使用这个双方协议与 CS 进行注册。最终，PC 端将随机产生的认证密钥 K、计算出的密文 F 以及用户凭证 C、$\tilde{C}$ 一起发送给 CS，CS 存储该信息状态。

3) store 模块：MP 存储 PC 端产生的数据信息 p_2 和 K_2，而 PC 端不需存储任何信息，用户只需记住（name，pwd）。

4) preAuth 模块：客户端 PC 使用其用户名 name 和口令 pwd，并借助 MP 从 CS 检索其密文信息 $\tilde{C}'$。CS 给 PC 发送 $\tilde{C}$ 和挑战信息 chal。

5) retrieve 模块：PC 客户端解密上述密文信息得到 K_1，并从 MP 中得到 K_2，此时 PC 客户端便可计算恢复出认证密钥 K。

6) authenticate 模块：PC 客户端使用 K 和 chal 向 CS 证明它有相应的账户凭证，CS 输出 accept 或者 reject。

4.2　安全游戏

在 SPASS 方案中，手机可能会丢失或者被盗，则恶意的对手可能使用用户存储在手机上的信息访问服务器，甚至破解用户的口令；PC 端和服务器端也可能会遭受到对手的攻击。因此，对 SPASS 方案定义 game one 和 game two 这 2 种安全游戏，由于用户通常只能记住一个简短的低熵口令，通常根据 2 个参数定义安全性[18]：k 是一个较大的数（80~128）bit，l 是一个较小的数（30~40） bit，用 $m<<k$ 来强调值 m 远远小于值 k。符号 ProbGuess(N)表示对手猜测 N 次可以猜到用户口令的概率。ProbGuess(N)表示认证方案的安全性的固有上界。定义 neg(k)：若对任意常数 c，存在有限的 K，对于任意 $k>K$，都有 neg(k)<k^{-c}，则 neg(k)是一个可以忽略的函数。

定义 1　猜测概率[18]　定义 Adv 是一个基于概率多项式时间（PPT）的对手，拥有用户在 userGen 阶段选取口令的先验知识。本文定义

$$\begin{aligned}&\mathrm{ProbGuess}(N)=\Pr[(\mathrm{name},\mathrm{pwd})\leftarrow\mathrm{UserGen}(1^k);\\&(\mathrm{pwd}_1^{'},...,\mathrm{pwd}_N^{'})\leftarrow\mathrm{Adv}(1^l,\mathrm{name}):\\&\mathrm{pwd}\in\{\mathrm{pwd}_1^{'},\cdots,\mathrm{pwd}_N^{'}\}]\end{aligned}\tag{1}$$

由于 pwd 的选择不需要服从均匀分布，所以 ProbGuess(N)$\geqslant\frac{N}{2^l}$。

定义 2　安全的口令加密[18]　若对于任意的 PPT 对手，adv 具有如下概率。

$$\begin{aligned}&\Pr[\mathrm{msg}\leftarrow\{0,1\}^k;(\mathrm{pwd}_0,\mathrm{pwd}_1,\mathrm{state})\leftarrow\mathrm{Adv}(1^k,1^l);\\&b\leftarrow\{0,1\};\mathrm{ctext}\leftarrow\mathrm{Encrypt}(\mathrm{pwd}_b,\mathrm{msg})\\&b'\leftarrow\mathrm{Adv}(1^k,\mathrm{state},\mathrm{ctext}):b=b']=\frac{1}{2}+\mathrm{neg}(k)\end{aligned}\tag{2}$$

则基于口令的加密方案就可以安全地阻挡随机消息下的字典攻击。

game one（蜜罐攻击）　在此游戏中，对手充当的是恶意的在线服务器，挑战者扮演的角色是 PC 端和手机，它们均被认为是可信的；而对手扮演服务器的角色，可以让 PC 客户端使用手机执行 register、preAuth 和 authenticate 阶段。若在这场游戏中，对手得到了用户口令，则对手胜利。

定义 3[18]　假设 PPT 对手赢得 game one 游戏的概率是 P_1，若 $P_1\leqslant\frac{1}{2}+\mathrm{neg}(k)$，则 SPASS 方案满足基于 game one 的安全性。

game two（外部攻击）　此游戏中，对手可以扮演 PC 客户端的角色，而挑战者扮演手机端和 N 个不同服务器的角色，对手将模拟 PC 客户端向每个服务器发起 T 次 PreAuth 和 authentication 请求。对手还可以扮演手机端的角色（即手机设备被对手偷掉），而挑战者扮演 PC 客户端和 N 个不同服务器的角色。如果对手可以在 authentication 阶段使服务器信服并输出 accept，即对手在此游戏的过程中猜到了正确的用户口令，则对手胜利。

定义 4[18]　假设 PPT 对手赢得 game two 游戏的概率是 P_2，若 $P_2\leqslant$ProbGuess(TN+1)+neg(k)，则

SPASS 方案满足基于 game two 的安全性。

定义 5 安全的 SPASS 方案 若 SPASS 方案可以同时满足基于 game one 的安全性和基于 game two 的安全性，则可以认为 SPASS 方案是安全的。

4.3 SPASS 方案的安全性

定理 SPASS 方案是满足定义 5 的安全方案。

证明

对于 game one 的安全游戏，场景可以还原如下[18]。

对手：即恶意的服务器，先选择 2 个口令 pwd_0 和 pwd_1，则挑战者将会在该区域选择位数 b，并设置 pwd_b 作为其口令。

Play：对手与客户端和手机端通过以下的协议进行交互。挑战者保留一个 sid，它可以唯一地标识每一次交互。

playRegister：PC 客户端必须与一个由对手选择的服务器进行注册。对手产生服务器名并且发送给挑战者。然后挑战者和对手执行 register，此时挑战者扮演的是诚实的 PC 客户端并输入（name，pwd，servername）。resister 协议完成后，挑战者模拟 PC 端和手机端之间的存储协议，存储（p_2，K_2）。

playPreAuth：对手要求 PC 端运行 PreAuth 协议。挑战者扮演 PC 端执行 PreAuth 协议，输入（name，pwd）。PreAuth 协议完成后，挑战者在它的 PreAuth 数据库存储（name，sid，chal）。

playAuthenticate：对手要求 PC 端运行 authenticate 协议。挑战者通过 PreAuth 数据库查看与 name 相关的 sid。随后挑战者模拟 PC 端和手机端之间的 retrieve 协议获得部分认证密钥 K_2，挑战者使用 K 和对手执行 authenticate 协议。

output 对手输出一个位数 b'，若 $b=b'$，则对手胜利。

由此可见，对手赢得此游戏仅仅需要区分 2 个口令。因此，对手猜测到正确口令的概率不可能大于 $\frac{1}{2}$。所以可以得出，对手能够赢得这场游戏的概率 $P_1 \leqslant \frac{1}{2}+\text{neg}(k)$，故 SPASS 方案具有基于 game one 的安全性。

对于 game two 的安全游戏，场景可以还原如下[37]。

1) 对手模拟客户端 PC，每猜测一个口令 pwd'，就向某个服务器发起一次执行 PreAuth 和 authentication 的请求，在每一次的交互阶段，对手都可以验证他猜测的口令是否正确，以便下一次做出更适合的猜测。对于每个服务器 i，$i\in[1,N]$，对手最多可以发起 T 次 PreAuth 和 authentication 请求，在该游戏的最终输出阶段，对手还可以多进行一次猜测，即最多进行 TN+1 次的猜测。因此，对手在该过程中猜测到用户口令的概率 P[对手得到口令]≤ProbGuess (TN+1)。故对手能够获赢得这场游戏的概率 $P_2 \leqslant$ ProbGuess (TN+1)+neg(k)。

2) 当对手扮演手机端的角色时，在协议的开始，手机是可靠的。对手不能得到 store 或 retrieve 请求的任何结果。在某一时刻，对手也许会损坏存储设备（手机）。一旦损坏手机，对手就会得到存储在手机上的所有信息。然而这时，挑战者会停止与设备的交互（模拟现实世界的场景：用户的手机一旦被窃取便被停止使用）。更加确切地说，挑战者不再模拟请求 store 或 retrieve，这意味着在 register 和 authentication 阶段的查询期间，挑战者将提前终止。因此，对手几乎不可能获得用户口令，故 P_2 是一个可以忽略的小值。

所以，SPASS 方案满足基于 game two 的安全性。

综上所述，SPASS 方案满足基于定义 3 和定义 4 的安全性，故 SPASS 方案是安全的。

证毕。

因此，用户完全不必担心手机丢失或遭受恶意软件侵袭，以及服务器端的蜜罐攻击和钓鱼网站的攻击，因为只能从手机上窃取部分随机串，而服务器端也不再直接存储用户的口令，或口令的确定函数。总之，任何一方的沦陷都不会影响用户私密信息（如口令）的安全。

5 性能分析与评估

本节主要从时间和存储方面对本方案的具体实用性进行评估。在时间方面，对 SPASS 方案的注册阶段和认证阶段分别在 3 种不同场景下进行所耗时间的测试，并对测试结果数据进行展示和对比分析。在存储方面，主要对 SPASS 方案具体实现过程中用户的参与设备，即 PC 端和手机端的存储量进行分析评估。

5.1 测试方案与场景

测试 SPASS 方案的实验设备为一台 PC、一个云服务器和一个安卓设备，其中的云服务器是租用部署于青岛的阿里云服务器，测试方案如图 4 所示。租用的云服务器的配置参数如表 1 所示。

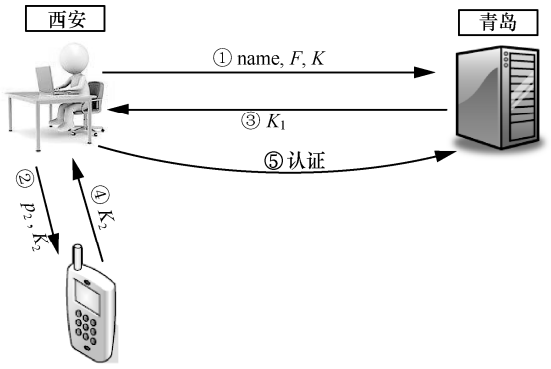

图 4　测试方案总体示意

表 1　　**云服务器参数**

服务商	硬盘	CPU	内存	操作系统	带宽
阿里云	40 GB	1 核	2 GB	Windows Sever2008 R2 企业版 64 位中文版	5 Mbit/s

客户端 PC 的配置参数如表 2 所示。

表 2　　**客户端 PC 参数**

服务商	硬盘	CPU	内存	操作系统
Dell Inc. Vostro 270	500 GB	4 核	4 GB	Windows7 64 位

安卓设备分为 2 类：1) 安卓模拟器 Android 4.3.1- API Level 18，CPU ARM(armeabi-v7a)，RAM 1024，VM Heap 32，Internal Storage 200 MiB；2) 安卓手机，魅族 MX4，Flyme OS 4.0，真八核处理器 MTK 6595 魅族定制版处理器；索尼 Xperia SL（LT26ii），Android 4.1.2，高通骁龙 MSM8260 双核 1.7 GHz 处理器。

SPASS 方案的测试程序主要在 Eclipse 的开发环境下由 JAVA 语言进行编写，方案中用到的 ElGamal、MAC 等的加密算法使用了 bouncy castle 库。测试场景如下。

场景 1　同一用户完成方案的注册阶段和认证阶段，输入相同的用户名和口令，分别测试整个注册阶段和认证阶段所需时间。各自测试 10 组数据，分别计算出用户在注册阶段和认证阶段所耗时间的平均值。

场景 2　分析口令熵对本协议运行时间的影响，因此在注册阶段和认证阶段分别选取 10 组不同用户，其设置的口令的长短、复杂度各不相同，最终，分别进行测试评估。

场景 1 和场景 2 中使用的安卓设备是安卓模拟器。

场景 3　分析不同的安卓设备（安卓模拟器、魅族手机、Sony 手机）对方案运行时间的影响。此场景中，用户分别使用这 3 种不同的设备完成方案的注册和认证阶段，分别测试其所耗时间。

方案的具体测试中需要的装备阶段：手机端和用户端已经分别和云服务器端建立认证信道，即它们均可以和服务器端交互认证信息，为了简化但又不失对协议性能评估的准确性，该阶段的性能不予测试。

因此，具体测试中将注册阶段的总时间 T_R（单位为 ms）分为 2 个部分：t_{R1} 和 t_{R2}。如图 5 所示，其中 t_{R1} 是从用户在 PC 端输入用户名 name 和口令 p 开始，到 PC 端生成 p 和认证密钥 K 的份额 p_1、p_2 和 K_1、K_2 以及计算出 p_1、K_1 的密文 F 结束时所耗时间；t_{R2} 是将用户的密文发送至云服务器所耗时间。

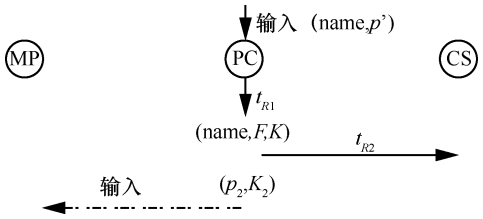

图 5　注册阶段流程说明

将认证阶段的总时间 T_A（单位为 ms）分为 6 个部分，分别是 t_{A1}、t_{A2}、t_{A3}、t_{A4}、t_{A5} 和 t_{A6}。如图 6 所示，其中 $t_{A1}+t_{A2}$ 为用户在 PC 端产生口令的密文，然后向云服务器端请求对应的认证密钥密文所耗时间之和；t_{A3} 为云服务器端进行加密计算及将密文发送给安卓设备所耗时间；t_{A4} 为安卓设备进行加密计算及将加密结果发送给云服务器端所耗时间；t_{A5} 为云服务器端验证口令及发送挑战和认证密钥密文所耗时间；t_{A6} 为 PC 端计算出用户的认证信息并发送给云服务器所耗时间。

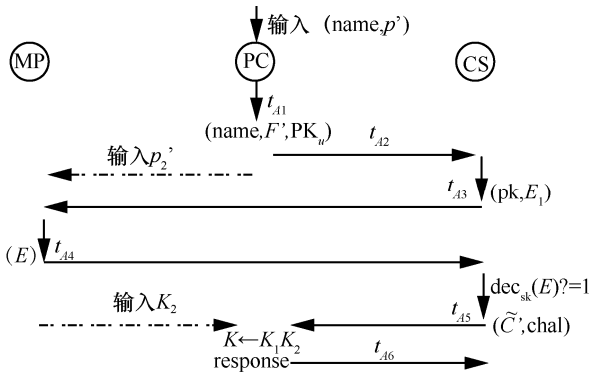

图 6　认证阶段流程说明

5.2 测试结果

5.2.1 场景 1 测试结果

1) 注册阶段

此场景下，同一用户在 PC 端输入相同的用户名和口令进行注册，重复进行 10 次，结果如表 3 所示。

表 3 场景 1 注册阶段时间/ms

次数	t_{R1}	t_{R2}	总时间 T_R
1	554.04	11.83	565.87
2	562.12	12.04	574.16
3	589.61	12.26	601.87
4	597.05	11.92	608.97
5	688.92	11.85	700.77
6	566.47	11.62	578.08
7	565.16	11.68	576.84
8	600.3	11.65	611.95
9	566.57	11.78	578.35
10	563.02	11.91	574.93

2) 认证阶段

用户想要访问服务器时，在 PC 端输入自己的用户名和口令，服务器对用户身份进行验证。分别测出认证阶段的 6 个部分所耗时间，将结果展示如表 4 所示。

5.2.2 场景 2 测试结果

情况 1 口令的长度不同

1) 注册阶段

选取 10 组不同的用户进行测试，用户在 PC 端输入其用户名和口令进行注册，其中不同用户所选口令的长度各不相同（例如，密码长度从一位增加至 10 位），测得的 10 组数据如表 5 所示。

表 5 场景 2 中的情况 1 注册阶段时间/ms

用户	t_{R1}	t_{R2}	总时间 T_R
user1	593.71	11.32	605.03
user2	660.72	11.43	672.15
user3	555.57	11.91	567.48
user4	633.15	11.41	644.56
user5	574.02	12.16	586.18
user6	571.38	11.66	583.04
user7	672.64	11.63	684.27
user8	564.88	11.52	576.40
user9	600.78	11.44	612.22
user10	563.25	11.39	574.93

2) 认证阶段

用户在 PC 端输入其已经注册的相应用户名和口令，测出当口令长度不同时认证阶段的各个部分所耗时间，测出的 10 组数据展示，如表 6 所示。

情况 2 口令复杂度不同

3) 注册阶段

在该情况下，选取 10 组不同用户进行测试。这 10 个用户分别在 PC 端输入其用户名和口令进行注册，其中不同用户的口令的复杂度各不相同（例如数字、大小写字母以及特殊字符的组合不同），测得的数据如表 7 所示。

4) 认证阶段

当用户需要登录服务器进行身份认证时，需要在 PC 端输入其已注册的用户名和口令，测出认证阶段的各个组成部分所耗时间，测出对应的 10 组数据，如表 8 所示。

表 4 场景 1 认证阶段时间/ms

次数	t_{A1}	t_{A2}	t_{A3}	t_{A4}	t_{A5}	t_{A6}	总时间 T_A
1	634.95	12.16	3 852.79	2 685.00	3 020.25	551.69	10 756.84
2	588.71	12.12	3 088.24	2 451.40	3 008.86	537.88	9 687.21
3	693.18	12.13	3 462.55	2 693.89	2 968.58	542.65	10 372.98
4	603.26	11.90	3 762.82	2 729.93	2 958.64	568.11	10 634.66
5	666.91	11.71	3 490.51	2 743.85	2 938.28	581.18	10 432.44
6	588.29	11.72	3 472.58	2 701.08	2 943.11	567.73	10 284.51
7	586.82	11.81	3 976.51	2 492.83	2 964.37	565.19	10 597.53
8	696.74	11.88	3 415.59	3 150.62	2 929.69	574.21	10 778.73
9	705.49	13.57	3 787.21	2 816.02	2 916.69	579.40	10 818.38
10	667.10	11.78	3 217.01	2 690.37	2 895.12	551.87	10 033.25

表 6　　场景 2 中的情况 1 认证阶段时间/ms

用户	t_{A1}	t_{A2}	t_{A3}	t_{A4}	t_{A5}	t_{A6}	总时间 T_A
user1	706.66	11.18	3 870.38	2 476.76	2 896.26	620.59	10581.83
user2	656.90	11.67	3 021.22	2 516.65	2 972.40	540.37	9719.21
user3	590.32	11.40	3 942.38	2 667.09	2 940.90	564.04	10716.13
user4	636.21	11.50	3 436.28	2 472.99	2 917.59	563.28	10037.85
user5	661.22	11.52	3 441.83	2 480.50	2 936.19	548.34	10079.60
user6	696.98	11.67	3 155.77	2 690.94	2 919.25	545.47	10020.08
user7	583.52	11.43	3 485.63	2 716.33	2 896.02	561.31	10254.24
user8	580.69	12.01	3 730.85	2 517.52	2 949.34	561.96	10352.37
user9	591.99	11.73	3 333.63	2 476.16	2 992.37	551.10	9956.98
user10	696.74	11.47	3 313.86	2 507.97	2 933.24	548.51	10011.79

表 7　　场景 2 中的情况 2 注册阶段时间/ms

用户	t_{R1}	t_{R2}	总时间 T_R
user1	576.09	11.99	588.08
user2	677.24	11.78	689.02
user3	553.82	11.67	565.49
user4	577.40	12.14	589.54
user5	561.34	11.67	573.01
user6	600.80	11.78	612.58
user7	659.23	12.15	671.38
user8	638.52	11.69	650.21
user9	578.69	11.68	590.37
user10	556.91	12.04	568.95

5.2.3　场景 3 测试结果

在此场景中，用户分别使用安卓模拟器、Sony 手机和魅族手机来对本协议的注册阶段和认证阶段进行测试。

1) 注册阶段

同一用户在 PC 端输入用户名和口令进行注册，对整个注册阶段所耗时间进行测试，如表 9 所示。

表 9　　场景 3 注册阶段时间/ms

设备	t_{R1}	t_{R2}	总时间 T_R
安卓模拟器	555.89	11.71	567.60
Sony	564.45	11.59	576.04
魅族	532.69	11.69	544.38

2) 认证阶段

用户使用这 3 种不同安卓设备分别参与认证时，在 PC 端输入自己的用户名和口令，请求服务器对其身份进行认证，测试此认证阶段的各个部分所耗时间，如表 10 所示。

表 8　　场景 2 中的情况 2 认证阶段时间/ms

次数	t_{A1}	t_{A2}	t_{A3}	t_{A4}	t_{A5}	t_{A6}	总时间 T_A
user1	674.89	11.56	3 589.07	2 487.21	2 682.28	539.95	10 291.97
user2	652.75	11.64	3 538.29	2 702.91	3 001.16	627.43	10 534.18
user3	598.73	11.67	3 730.33	2 710.31	3 160.28	524.04	10 735.36
user4	597.75	11.60	3 389.03	2 670.76	2 921.73	532.14	10 123.01
user5	602.49	11.78	3 558.93	2 792.32	3 042.90	543.13	10 551.55
user6	679.08	12.04	3 584.63	2 780.92	2 851.92	583.65	10 492.24
user7	705.12	11.57	3 401.68	2 665.76	2 845.77	550.11	10 180.01
user8	602.14	11.81	3 769.10	2 786.24	2 858.46	562.39	10 590.14
user9	627.65	12.54	3 329.07	2 664.68	2 886.40	558.95	10 079.29
user10	611.41	11.89	3 564.91	2 682.28	2 844.04	529.52	10 244.05

表 10 场景 3 认证阶段时间/ms

设备	t_{A1}	t_{A2}	t_{A3}	t_{A4}	t_{A5}	t_{A6}	总时间 T_A
安卓模拟器	655.94	12.14	3 746.96	2697.21	2 886.60	603.58	10 602.43
Sony	631.69	11.53	3 737.36	174.27	2 911.79	598.03	8 064.67
魅族	627.51	11.68	3 758.87	152.76	2 919.66	574.56	8 045.04

5.3 性能分析

根据场景 1 的注册阶段所测数据结果，如表 3 所示，可以计算出用户在 PC 端完成注册时所需要的平均时间大约为 597.18 ms，如图 7 所示。由认证阶段所测数据结果，如表 4 所示，可计算出用户使用安卓模拟器时，实现服务器对 PC 用户身份进行认证时所需要的平均时间为 10 439.65 ms，如图 8 所示。

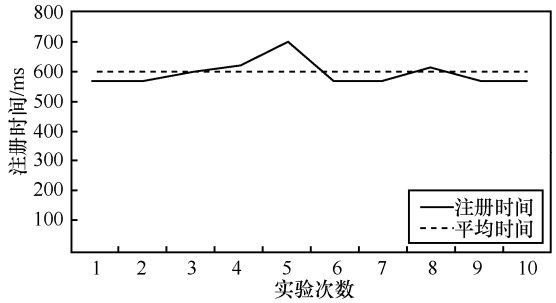

图 7 场景 1 注册阶段实验数据和平均时间

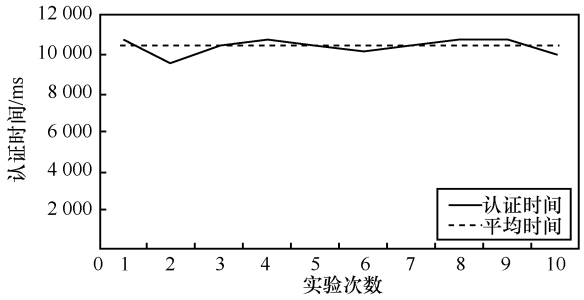

图 8 场景 1 认证阶段实验数据和平均时间

由场景 2 中情况 1 所测得的注册阶段数据结果，如表 5 所示，可以得出，当不同的用户输入长度不同的口令进行注册时，这个注册阶段所耗费的最长时间与最短时间相差 116.79 ms，小于相同口令的差值 134.9 ms，故口令的长度对本协议注册阶段所耗时间影响不大。可画出长度不同的用户口令在注册阶段的耗时对比，如图 9 所示。

由场景 2 中情况 1 所测得的认证阶段数据结果，如表 6 所示，可以计算出当不同用户输入其对应的口令进行认证时，所耗费的最长时间与最短时间的差值为 996.92 ms，小于相同口令的差值 1 131.17 ms，故口令的长度对本协议的认证阶段总时间的影响不大。图 10 为不同长度的用户口令在认证阶段的耗时对比。

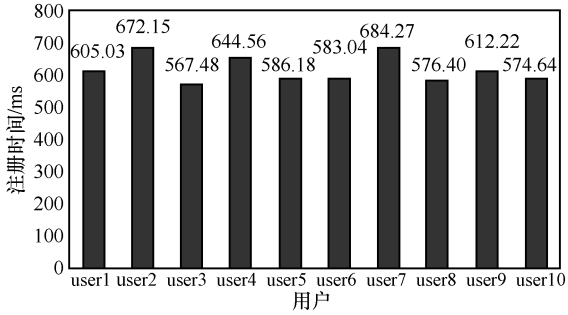

图 9 口令长度不同时注册阶段实验数据

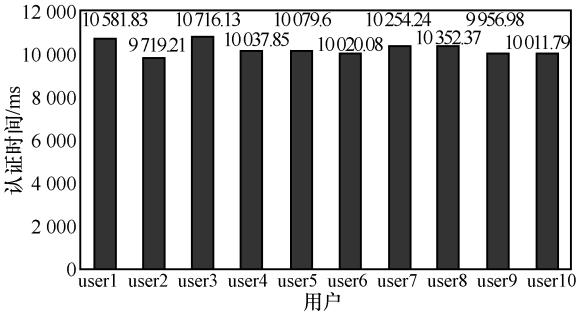

图 10 口令长度不同时认证阶段实验数据

根据场景 2 中情况 2 的注册阶段所测得的数据，如表 7 所示，可以得出，当口令的复杂度不同时，所耗费的最长时间和最短时间相差 123.53 ms，同样，小于相同口令的差值 134.9 ms，所以说口令的复杂度对本协议的注册阶段的耗时影响不大。此情况的注册阶段的耗时对比如图 11 所示。

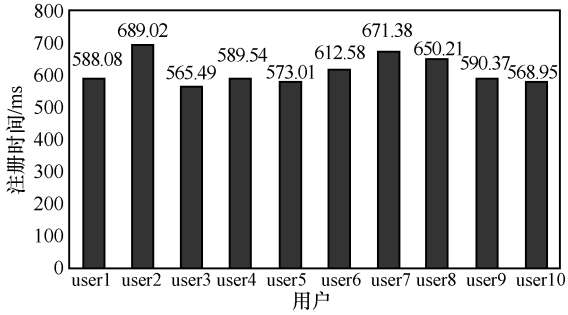

图 11 口令复杂度不同时注册阶段实验数据

根据场景 2 中情况 2 的认证阶段所测得的数据（如表 8 所示）可以得出，当不同用户输入其对应的口令认证时，所耗费的最长时间与最短时间的差值是 656.07 ms，远小于相同口令时的差值 1 131.17 ms，所以口令的复杂度对认证总耗时影响不大。此情况下，不同用户在认证阶段的耗时对比如图 12 所示。

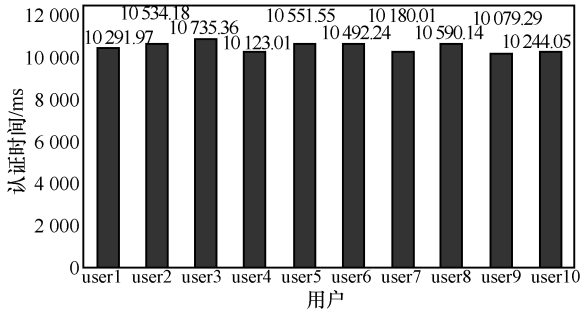

图 12　口令复杂度不同时认证阶段实验数据

场景 3，即用户使用不同的安卓设备参与认证，由表 9 测得的数据可以看出，3 种设备在整个注册阶段的总耗时并没有太大差异（最多相差 32.22 ms），这是因为本方案的注册阶段是在 PC 端和服务器之间完成的，安卓设备只是被用来存储一些用户数据信息。而由表 10 所展示的数据结果可以看出，在协议的认证阶段，使用安卓模拟器参与协议认证所消耗的总时间远远大于使用实际手机参与认证所耗费的总时间，而两者在实际中所耗费的传输时间几乎没有差别，因此可以得出，这种明显的时间差距主要是来自安卓设备的计算时间。协议中使用不同安卓设备参与认证时，注册阶段和认证阶段所耗时间对比如图 13 所示。

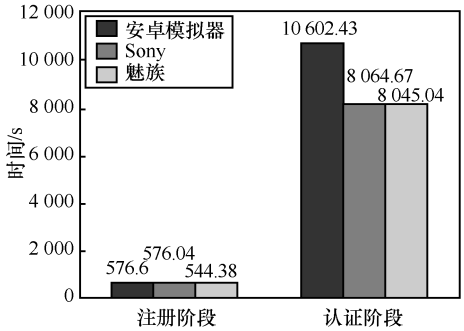

图 13　场景 3 中 2 个阶段实验数据对比

由测试方案可知，安卓设备的计算时间和传输时间总计为t_{A4}，但由于安卓模拟器与服务器之间的传输时间和实际手机设备与服务器的传输时间基本无差别（小于 3 ms），所以在安卓设备上的计算时间就可以使用t_{A4}进行对比，用户在不同的安卓设备上进行计算所耗时间对比如图 14 所示。由图 14 可以看出，本协议的认证阶段，不同类型的安卓设备的计算耗时中，安卓模拟器的最长，而 2 个安卓手机（Sony 手机和魅族手机）的计算耗时差距不大，产生这种差距的主要原因是安卓模拟器在处理器和设置参数上与安卓手机相比有一定的差距，所以影响了其运行效率。

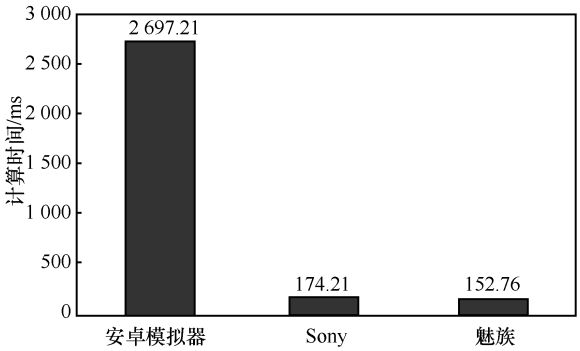

图 14　不同设备算法运行时间对比

由于已有的 PPSS[21]和 2PASS[22]方案都旨在解决如何在不可信的服务器上安全存储用户个人数据信息的问题，并没有涉及关于用户和多个在线服务进行的单一口令安全认证。因此，本文的方法仅与已有的 mobile SPA[20]进行性能比较。表 11 给出了已有的 mobile SPA[20]方案和本文的 SPASS 方案在注册阶段个人电脑（PC）与移动智能手机（MP）的主要计算消耗对比，其中，mac 为 MAC 运算，h 为散列运算，s 为加密/解密运算。已知 mobile SPA 方案在认证阶段的时间消耗是其在云服务器（CS）的计算时间为 1 ms、在 MP 的计算时间为 2 ms 和传输时间为 35 ms 的总和，即 38 ms；而本文的 SPASS 方案在手机上的测量时间都高于 38 ms。由此可见，仅在计算消耗方面，本文的方案高于 mobile SPA 方案。

表 11　　计算消耗对比

设备	mobile SPA	SPASS
PC	1mac+1h+1s	1mac+3s
MP	0	0

但是，本文的 SPASS 方案对认证密钥和口令进行了编码、随机取串、签名运算以及零知识证明等，

大大提高了方案的安全可靠性，在实际的使用过程中能够明显体现出来。表 12 给出了 mobile SPA 方案与本文提出的 SPASS 方案部分简单常见的安全指标对比。

表 12 安全指标对比

安全指标	mobile SPA	SPASS
双向认证	是	是
抗口令丢失攻击	弱	强
抗中间人攻击	弱	强
抗字典攻击	弱	强
抗恶意软件攻击	弱	强
抗服务器内部攻击	弱	强

在存储方面，PC 端不需要存储任何信息，因此用户可以使用不同的客户端电脑来访问其账户。而用户需要通过对不同的服务器使用不同的认证密钥 K 来防止该服务器模仿自身登录访问其他服务器，因此，移动设备的存储量应该和服务器的数量是线性关系。一个 MAC 密钥是 128 B，一个内存为 1MB 的移动设备就可以存储 8 192 个服务器的认证信息。因此，可使用当前标准的移动设备实现线性存储。

综上所述，本文提出的方案在时间性能方面虽然没有什么明显的优势，尤其是认证阶段耗时略长，但是在存储及安全方面却有很大的提高。

6 结束语

当用户使用单一口令和多个在线服务进行认证时，为了保障口令的安全性和用户身份认证方法的完善性，本文提出了 SPASS 方案利用手机辅助用户认证，其中 PC 端不存储用户的任何认证信息，且认证信息并非单一地存储于手机端，而是在手机端和服务器端共享。此外，经过一系列的计算，手机端和服务器端存储的认证信息均为随机串，即使任何一方遭受攻击，攻击者也只可能恢复部分认证信息，而无法获得完整的认证信息。经过严格的安全性证明和实验测试，结果表明，SPASS 方案虽然在时间效率上没有太大的优势，但是具有较高的安全性能，可以在远程用户仅使用单一口令和多个在线服务进行认证时，保护用户口令不会受到字典攻击、蜜罐攻击等威胁，即使在成功的中间人攻击下，也可以保护用户固定且长期的口令，具有很高的应用价值。

参考文献：

[1] FLORENCIO D, HERLEY C. A large-scale study of Web password habits[C]//Proceeding of the 16th international conference on World Wide Web. 2007, ACM, 2007: 657-666.

[2] CARSON N. In: 2004, Mark Zuckerberg broke into a facebook user's private email account[EB]. Business Insider, 2010.

[3] BELLOVIN S M, MERRITT M. Encrypted key exchange: password-based protocols secure against dictionary attack[C]//Computer Society Symposium on Research in Security and Privacy. 1992: 72-84.

[4] WU T D. The secure remote password protocol[C]//The Network and Distributed System Security Symposium.1998, 98: 97-111.

[5] JABLON D P. Strong password-only authenticated key exchange[J]. ACM SIGCOMM Computer Communications Review, 1996, 26(5): 5-26.

[6] BELLOVIN S M, MERRITT M. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise[C]//The 1st ACM Conference on Computer and Communications Security. ACM, 1993: 244-250.

[7] GENTRY C, MACKENZIE P, RAMZAN Z. A method for making password-based key exchange resilient to server compromise[C]//Annual international Cryptology Conference. Springer Berlin Heidelberg, 2006: 142-159.

[8] BOYEN X. Hidden credential retrieval from a reusable password[C]//Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. ACM, 2009: 228-238.

[9] BOYEN X. Hpake: password authentication secure against cross-site user impersonation[C]//International Conference on Cryptology and Network Security. 2009: 279-298.

[10] JUNG J, LEE D, KIM J, et al. Cryptanalysis and improvement of efficient password-based user authentication scheme using hash function[C]//The 10th International Conference on Ubiquitous Information Management and Communication. 2016: 23.

[11] WEI J, LIU W, HU X. Secure and efficient smart card based remote user password authentication scheme[J]. International Journal of Network Security, 2016, 18(4): 782-792.

[12] TSAI C Y, PAN C S, HWANG M S. An improved password authentication scheme for smart card[C]//International Conference on Intelligent and Interactive Systems and Applications. 2016: 194-199.

[13] OM H, BANERJEE S. A password authentication method for remote users based on smart card and biometrics[J]. Journal of Discrete Mathematical Sciences & Cryptography, 2017, 20(3): 595-610.

[14] GIRI D, SHERRATT R S, MAITRA T. A novel and efficient session spanning biometric and password based three-factor authentication protocol for consumer USB Mass Storage Devices[J]. IEEE Transactions on Consumer Electronics, 2016, 62(3): 283-291.

[15] 李晓伟, 杨邓奇, 陈本辉, 等. 基于生物特征和口令的双因子认证

与密钥协商协议[J]. 通信学报, 2017, 38(7): 89-95.
LI X W, YANG D Q, CHEN B H, et al. Two-factor authenticated key agreement protocol based on biometric feature and password[J]. Journal on Communications, 2017, 38(7): 89-95.

[16] WU M, GARFINKEL S, MILLER R. Secure web authentication with mobile phones[J]. Dimacs Workshop on Usable Privacy & Security Software, 2004.

[17] 安迪, 杨超, 姜奇, 等. 一种新的基于指纹与移动端协助的口令认证方法[J]. 计算机研究与发展, 2016, 53(10): 2400-2411.
AN D, YANG C, JIANG Q, et al. A new password authentication method based on fingerprint and mobile phone assistance[J]. Journal of Computer Research and Development, 2016, 53(10): 2400-2411.

[18] MCCUNE J M, PERRIG A, REITER M K. Seeing-is-believe: using camera phones for human-verifiable authentication[C]//Security and Privacy, 2005 IEEE Symposium on. IEEE, 2005: 110-124.

[19] STARNBERGER G, FROIHOFER L, GOESCHKA K M. QR-TAN: secure mobile transaction authentication[C]//International Conference on Availability, Reliability and Security. 2009: 578-583.

[20] ACAR T, BELENKIY M, KÜPÇÜ A. Single password authentication[J]. Computer Networks,2013, 57(13): 2597-2614.

[21] BAGHERZANDI A, JARECKI S, SAXENA N, et al. Password-protected secret sharing[C]//The 18th ACM conference on Computer and Communications Security. 2011: 433-444.

[22] CAMENISCH J, LYSYANSKAYA A, NEVEN G. Practical yet universally composable two-server password-authenticated secret sharing[C]//Proceedings of the 2012 ACM Conference on Computer and Communications Security. 2012: 525-536.

[作者简介]

徐渊（1991− ），女，陕西西安人，西安财经大学助理实验师，主要研究方向为云计算、网络安全。

杨超（1979− ），男，陕西西安人，博士，西安电子科技大学教授、博士生导师，主要研究方向为密码学、信息和网络安全。

杨力（1977− ），男，陕西西安人，博士，西安电子科技大学教授、博士生导师，主要研究方向为移动互联网安全、云计算安全和可信计算技术。