Two Birds with One Stone: Two-Factor Authentication with Security Beyond Conventional Bound

Ding Wang, *Student Member, IEEE* and Ping Wang, *Senior Member, IEEE*

Abstract—As the most prevailing two-factor authentication mechanism, smart-card-based password authentication has been a subject of intensive research in the past two decades, and hundreds of this type of schemes have wave upon wave been proposed. In most of these studies, there is no comprehensive and systematical metric available for schemes to be assessed objectively, and the authors present new schemes with assertions of the superior aspects over previous ones, while overlooking dimensions on which their schemes fare poorly. Unsurprisingly, most of them are far from satisfactory—either are found short of important security goals or lack of critical properties, especially being stuck with the security-usability tension. To overcome this issue, in this work we first explicitly define a security model that can accurately capture the practical capabilities of an adversary and then suggest a broad set of twelve properties framed as a systematic methodology for comparative evaluation, allowing schemes to be rated across a common spectrum. As our main contribution, a new scheme is advanced to resolve the various issues arising from user corruption and server compromise, and it is formally proved secure under the harshest adversary model so far. In particular, by integrating "honeywords", traditionally the purview of system security, with a "fuzzy-verifier", our scheme hits "two birds": it not only eliminates the long-standing security-usability conflict that is considered intractable in the literature, but also achieves security guarantees beyond the conventional optimal security bound.

Index Terms—Two-factor authentication, smart card loss attack, Zipf's law, provable security, random oracle model

1 Introduction

Among the numerous methods for user access control, password-based authentication is the most widely used and acceptable mechanism because of its easy-operation, scalability, compatibility and low-cost advantages [1]. In such authentication schemes (some notable ones include SRP [2], KOY [3] and J-PAKE [4]), each user is assumed to only hold a memorable, low-entropy password, while the server needs to store a password-related verifier table necessary to verify the authenticity of users.

An inherent limitation of these password-only mechanism is that, the server has to *store a sensitive verifier table* that contains the passwords of all the registered users. Even if passwords are properly stored in salted-hash, once the authentication server is compromised, an overwhelming fraction of users' passwords will be exposed (see [5]) due to two reasons: (1) Human-beings' memory is inherently limited/stable, and the distribution of user-chosen passwords are highly skewed [6]; and (2) Password cracking hardware (e.g., GPUs) and algorithms (e.g., Markov-Chain-based [7]) are constantly being improved. At Password'12, Gosney [8] showed that a rig of 25 GPUs can test up to 350 billion guesses per second in an offline dictionary attack against traditional hash functions (e.g., NTLM and MD5). More sophisticated password hash functions (e.g., bcrypt and PBKDF2) only provide some relief [9], but with the cost of an honest server increasing by the same factor with the attacker, while the attacker is likely to be better equipped with dedicated password-cracking hardware.

These days it is no news to hear that millions of user accounts are breached in an on-line hacking incident. Some quite recent password data breaches include Adobe (150 million), Evernote (50 million), Anthem (40 million) [10], Rockyou (32 million) [11], Tianya (30 million), Dodonew (16 million), 000webhost (15 million) [12], Gmail (4.9 million) and Phpbb (255 K), just to name a few. Some services (e.g., Anthem [10] and Phpbb [13]) even have been breached more than once during the last five years. What makes things worse is that, users tend to reuse the same password (or slight variations) to access multiple servers (e.g., 43-51 percent of users as reported in [14]), a compromise of one server will lead to the failure of all other servers, which is described as the "domino effect" of password re-use.

To address the issue of password leakage from a compromised server, threshold password-only authentication schemes (e.g., [15]) have recently been proposed. In such schemes, the password files and user data are distributed over multiple severs, and thus no coalition of servers up to a certain threshold can learn anything about the password. However, they are inherently unable to cope with password leakages at the user side (e.g., hidden camera, key-loggers and phishing). To overcome this problem, leakage-resilient password systems (LRPS) [16] have been advocated. In such schemes, a user needs to input the password indirectly and this imposes an extra burden on common users.

• *The authors are with the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, P.R. China. E-mail: {wangdingg, pwang}@pku.edu.cn.*

Manuscript received 8 June 2016; revised 22 Aug. 2016; accepted 28 Aug. 2016. Date of publication 1 Sept. 2016; date of current version 6 July 2018. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TDSC.2016.2605087

Fig. 1. Smart-card-based password authentication.

Recently, it has been revealed [17] that, to attain both reasonable security and acceptable usability, LRPS schemes have to employ certain trusted devices to well address the security threat of password leakage on user side.

The limitations of threshold and LRPS schemes entrench the third approach—introduce smart cards as "a second line of defense". This gives rises to the smart-card-based password authentication, often termed as "two-factor authentication". This kind of authentication was proposed as early as about thirty years ago [18], and it has been widely deployed for various kinds of security-critical applications, such as e-commerce, e-banking and e-health. It also constitutes the basis of three-factor authentication [19], [20]. The participants of this sort of authentication (see Fig. 1) mainly involve a client U and an authentication server S [21], [22]. At first, U registers to S by submitting her self-chosen credentials (e.g., her identity and password) to S, then S securely issues U a smart-card with some security parameters. This is the user registration phase. Later on, U and S authenticate themselves to each other through the login phase. Besides, U may regularly change her password via the password change phase. Note that, protocols that leverage short message service as the second authentication channel and need to cooperate with the telecommunication service provider are out of the scope of this paper.

The most essential security goal of smart-card-based password authentication schemes is to achieve "truly two-factor security" [21], which means that only the user who is in possession of both a smart card and the corresponding password can login the service server. That's to say, a truly two-factor scheme shall be able to satisfy the following requirements: (1) an attacker in possession of a user's smart card (and able to extract the contents of the card) should not be able to perform an off-line dictionary attack to recover the user's password or impersonate the user; and (2) an attacker who learns a user's password, but does not get this user's smart card, should not be able to impersonate the user. Besides two-factor security, a practical scheme should also be able to withstand various passive and active attacks [23], [24], such as stolen-verifier attack, denial of service attack, reflection attack and parallel session attack. In addition, it is desirable that schemes can support some important properties [25], [26] like local password change, session key agreement and user anonymity.

There have been hundreds of works dealing with two-factor authentication in recent years (some notable ones include [22], [27], [28], [29]). However, in most studies the authors present attacks on previous schemes and propose new protocols with assertions of the superior aspects of their schemes, while ignoring the features that their schemes fail to provide, thus overlooking dimensions on which their schemes fare poorly. As such, *fair comparison and general consensus are unlikely*. Another common feature of these studies is that, *there is no proper security justification* (let alone an explicit security model) presented, which explains why these protocols previously claimed to be secure eventually turn out to be vulnerable. The research history of this area falls into the unsatisfactory cycle:

New protocol→broken →improved protocol →
broken again →further improved protocol → ⋯ .

For a more concrete grasp, we summarize the "break-fix-break-fix" history of two-factor authentication in Fig. 2. Note that many other important schemes cannot be incorporated into the figure only because of space constraints.

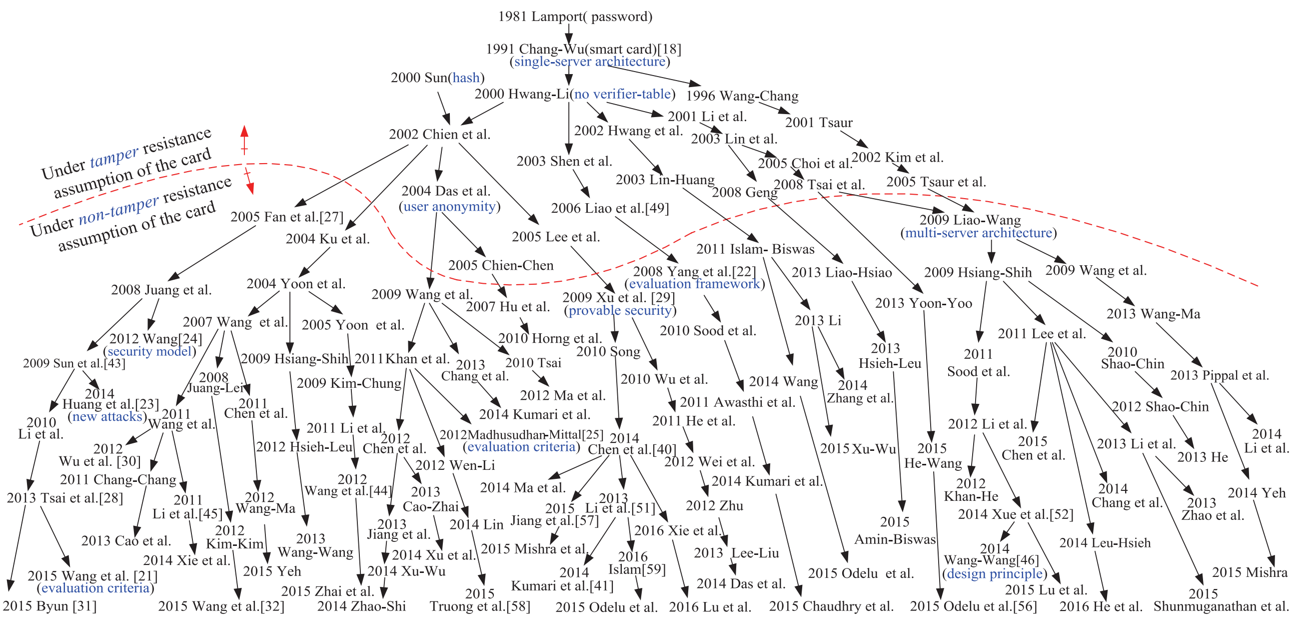

Fig. 2. A brief history of smart-card-based password authentication. Most works follow a similar research pattern: Each first proposes practical attacks on scheme(s) in the parent node and then suggests an improved scheme. The seminal contributions of some works are shown in corresponding parentheses. This "break-fix-break-fix" cycle has generated a wealth of literature but no much research progress.

1.1 Motivations

Though considerable efforts have been devoted to the development of secure and efficient two-factor authentication schemes, yet no much progress has been made. This is best illustrated by the recent somewhat surprising revelation [21] that "certain goals are beyond attainment" by just using existing cryptographic design techniques, which means all these previous attempts have failed in vein. A lot of literature has been generated, while little attention has been paid to the *systematic* design and evaluation of this sort of schemes and as a result, there is no common basis that allows schemes to be assessed thoroughly and fairly. As such, more 'improved schemes' essentially mean more tangles being added to the already tangled world. This well explains the long-standing failures in getting a secure and efficient two-factor authentication scheme.

As shown in Fig. 2, numerous 'improvements' have been proposed, however, most of them have been shortly found either unable to meet some important security goals or short of a few critical features. The crux lies in how to achieve the following two goals simultaneously: (1) truly two-factor security even if the smart cards may be lost and tampered; and (2) local and secure password update. This crux was left as an open problem by Huang et al. in 2014 [23]. Unfortunately, Wang et al. [21] recently find that, under the current two-factor cryptographic protocol design techniques, this problem is highly likely to be intractable. Are there techniques (e.g., ones from the system security domain) beyond the known cryptographic approaches can be used as a hedge against the failure (due to low-entropy passwords and conditional tamper resistance of smart cards) of conventional cryptographic protections?

Another unsatisfactory aspect of the existing literature is that, when evaluating the security guarantees of a two scheme (e.g., see some notable schemes in [22], [29], [30]), user-chosen passwords are invariably assumed to be *uniformly* drawn from the password space $\mathcal{D}$. Since this assumption is far from realistic, it may give rise to great *misconceptions* of the actual security that a scheme can provide. Under this assumption, even if the parameters in the smart card have been extracted by $\mathcal{A}$, the probability of $\mathcal{A}$'s success in one online guessing attempt is precisely $1/|\mathcal{D}|$. What a secure two-factor protocol $\mathcal{P}$ can assure is that, active online guessing is the best that $\mathcal{A}$ can do and other attack vectors (e.g., offline password guessing, replay and parallel session) are of little help. More specifically, this means $\mathcal{A}$'s optimal advantage (see [22], [29], [30], [31], [32] for example) in attacking $\mathcal{P}$ is no larger than $q_{send}/|\mathcal{D}| + \epsilon$, where q_{send} denotes the number of online impersonation attempts that $\mathcal{A}$ engages in and ϵ denotes a negligible value. However, user-chosen passwords are far from uniformly distributed and actually, they are on the other extreme. As shown in Section 5.3, on average $\mathcal{A}$ would gain an advantage of 3.33, 4.28, 8.21, 15.09 percent in just 3, 10, 10^2 and 10^3 online guessing attempts, respectively; but not the assumed advantage of $3/10^6$, $10/10^6$, $10^2/10^6$, $10^3/10^6$,[1] respectively. This misconception about the realistic security guarantee that a scheme can provide is evidently undesirable—*the actual level of security risk of $\mathcal{P}$ is largely underestimated*.

1. Note that the size of user password space $\mathcal{D}$ increases as the user base increases, and thus it is not a constant (see Section 5.3). It is generally assumed to be about $2^{20} \approx 10^6$ [33] as a rule of thumb.

1.2 Our Contributions

In this work, we take a first substantial step towards breaking the "break-fix-break-fix" fast knot by investigating into the underlying adversary model and by eliminating the deficiencies (e.g., insufficiencies, ambiguities and redundancies) in the current evaluation criteria set. We explicitly characterize the practical capabilities of an adversary, and suggest a broad set of 12 independent criteria framed as a systematic methodology for comparative evaluation. Though not cast in stone, it is expected that this list of requirements and their specific definitions provide a solid basis to work on. By introducing defensive techniques from the system-security domain, we advance a simple, robust and efficient scheme that addresses the long-standing security-usability conflict and achieves security beyond the conventional optimal security bound $q_{send}/|\mathcal{D}| + \epsilon$.

In summary, our contributions are three-fold:

1) First, we suggest a systematic framework for evaluating two-factor authentication schemes. It is composed of a practical adversary model as well as a well-refined criteria set. As far as we know, the adversary model is the harshest one to date, and the criterion set is more concrete and comprehensive as compared to related works. The effectiveness and practicality of this framework is demonstrated and tested by rating 67 typical two-factor schemes. It is expected to help facilitate better assessment of current and future schemes.
2) Second, we for the first time introduce the defensive tactic of "honeywords" [34], traditionally the purview of system security, into cryptographic protocol design. By integrating "honeywords" with our proposed "fuzzy-verifiers", our scheme can timely detect user card corruption to thwart online guessing and well address the seemingly intractable security-usability issue left in [23]—"whether or not there exist secure smart-card-based password authentication schemes and the password-change phase does not need any interaction with the server"?
3) Third, we show that our scheme can satisfy all the 12 criteria in our evaluation framework and be formally proved secure in the random oracle model under the harshest adversary model so far. In particular, we use large-scale real-life passwords to demonstrate the effectiveness of our integration of "honeywords" with a "fuzzy-verifier". We also show that our integration technique is generic and can be readily applied to two-factor schemes for various other environments (besides the client-server architecture examined in this work).

2 Attacker Model and Evaluation Criteria

We now explicitly define a realistic adversary model and present a set of twelve properties framed as a systematic methodology for comparative evaluation. They *together* make it possible for schemes to be rated across a common spectrum. Here we take inspiration from Bonneau et al.'s framework [1] for evaluating web authentication schemes based on usability and security principles.

2.1 Adversary Model

In the conventional password authenticated key exchange (PAKE) protocols, the attacker $\mathcal{A}$ is modeled to have full control of the communication channel between the communicating parties [35], [36], such as eavesdropping, intercepting, inserting, deleting, and modifying any transmitted messages over the public channel. To characterize forward secrecy, $\mathcal{A}$ may also be allowed to corrupt valid parties to attain long-term private keys. Besides, previous session key (s) may be obtained by $\mathcal{A}$ because of a number of reasons (e.g., improper erasure).

Recent studies have reported that, the secret parameters stored in common smart cards could be extracted (or partially extracted) by power analysis attacks [37], the software loophole exploiting attacks (launched on software-supported card, e.g., Java Card) [38] or reverse engineering techniques [39]. Consequently, the leakage of sensitive parameters stored in the smart card may lead the originally secure schemes vulnerable to the smart card loss problem, such as offline password guessing attack (e.g., the problematic schemes in [28], [40]) and impersonation attack (e.g., the problematic schemes in [29], [41]). Consequently, it is more prudent and desirable to design two-factor schemes under the assumption that the secret keys stored in the smart card could be revealed by some means. What's more, as observed and in-depth investigated by Wang [24] quite recently, malicious card readers also contribute to the security failures of such schemes. Once the card reader is under the control of the attacker (e.g., the card reader is infected with viruses and/or Trojans), the card owner's input password may be intercepted.

However, we restrict the attacker from first intercepting the password via the malicious card reader and then reading the information stored in the card via side-channel attacks. Otherwise, this combination will enable the attacker to trivially break any two-factor authentication protocols. This treatment adheres to "the extreme-adversary principle" [42]: Robust security is to protect against an extremely powerful adversary, of whom the only restricted powers are those that would allow her to trivially break any of this type of schemes. Moreover, this treatment is reasonable in reality: (1) the user is at the scene when she inserts her card into a malicious terminal, and there is little chance for the attacker to launch side-channel attacks (which needs special instruments and attack platforms); (2) the attacker is unlikely to succeed in revealing the sensitive data on the card within a short period of time.

All this implies that the common non-tamper-resistance assumption made about the smart cards shall be *conditional*, i.e., only when the card might be in the attacker's hands for a relatively long time (e.g., the card is lost/stolen), while in the other scenarios (e.g., the user inserts her card into a malware-infected card reader), the card remains tamper-proof. However, if a memory USB stick is used in such an untrusted terminal, both the user's password and the data stored in the card memory will be exposed easily without incurring any abnormality. This well explains the essential advantage of using smart cards over employing common cheap memory sticks, even if (conditional) non-tamper resistance assumption of the smart cards are made and smart cards are more expensive than memory sticks.

TABLE 1
Capabilities of the Adversary

C-01	$\mathcal{A}$ can offline enumerate the Cartesian product $\mathcal{D}_{id} \times \mathcal{D}_{pw}$.
C-02	$\mathcal{A}$ can determine the victim U_i's identity ID_i.
C-1	$\mathcal{A}$ has full control of the communication channel.
C-2	$\mathcal{A}$ may either (i) learn the victim's password via malicious card readers, or (ii) extract the secret data in the card by side-channel attacks, but cannot realize both. Otherwise, it is a trivial case.
C-3	$\mathcal{A}$ can learn the previous session key(s).
C-4	$\mathcal{A}$ can learn S's long-time private key(s) as well as all other data stored in S only when evaluating the eventual failure of S (e.g., known key attack and forward secrecy).

Our above analysis also invalidates the overly conservative proposition [22], [43] that "we simply consider a smart-card to be a memory card with an embedded micro-processor for performing required operations specified in a scheme." and "we put aside any special security feature that could be supported by the smart card". As memory-card-based schemes are completely insecure when used in un-trusted terminals, all the schemes based on such an extreme assumption like that of [22], [43] are as insecure as memory-card-based schemes, and they can never provide truly two-factor security when used in un-trusted terminals. Therefore, our "conditional" non-tamper resistance assumption of the smart cards is more reasonable than the extreme assumption like that of [22], [43].

Furthermore, for the sake of user-friendliness, a user is often allowed to select her own identity *ID* at will (maybe confined to a predefined format) during the registration phase; the user usually tends to choose an easy-to-remember identity which is of low entropy. Thus, user identities can also be offline enumerated by $\mathcal{A}$ within polynomial time. Hence, in practice, it is realistic to assume that $\mathcal{A}$ can offline enumerate all the (ID, PW) pairs in the Cartesian product $\mathcal{D}_{id} \times \mathcal{D}_{pw}$ within polynomial time. In contrast, many schemes that attempt to preserve user anonymity (e.g., [41], [44]) explicitly assume $\mathcal{A}$ cannot guess both *ID* and *PW* correctly at the same time. Such schemes may be problematic under our assumption.

The capabilities of the adversary $\mathcal{A}$ in our model is summarized in Table 1. Our work, following [22], [24], [45] while providing new insights, is among the few ones that explicitly specify $\mathcal{A}$'s capabilities. Since a protocol can only be 'secure' under some specific security model, it is hardly able to fairly evaluate the goodness of a scheme if no security model is given. In 2012, Wang [24] presented three kinds of security models, namely Type I, II and III. The Type-III model is the most powerful one to date, and it mainly makes three assumptions:

1) $\mathcal{A}$ has full control of the communication channel, which is consistent with our C-1 in Table 1;
2) The smart card is assumed to be non-tamper resistant and the user's password may be intercepted by $\mathcal{A}$ using a malicious card reader, but not both, which is consistent with our C-2 in Table 1;

3) The smart card has no counter protection, i.e., $\mathcal{A}$ can issue a large amount of queries to the card using a malicious card reader to learn useful information.

With regard to Assumption 3, we argue that this assumption may not be of much practical significance, because whether it is valid or not in practice has little relevance with protocol security under Assumption 2. On the one hand, if there is no verification of the input password before the run mode of the smart card, the only way that $\mathcal{A}$ can learn some useful information (except the static data stored in the card, which can be learnt by $\mathcal{A}$ under Assumption 2) is to interact with the remote server, which can be effectively thwarted by the server, e.g., locking the corresponding user account after a few failed login attempts. On the other hand, if this verification exists, $\mathcal{A}$ can always find the password that passes the verification by exhaustively inputting her guessing passwords into the malicious card reader, and with Assumption 2, secret data can also be extracted out. This is explicitly not allowed in the Type III model. Hence, Assumption 3 is not considered in our model. As with [44], we may simply assume that there is counter protection in the card, i.e., the card will be locked for a time period if the query number exceeds a certain threshold (e.g., the GSM SIM card V2 or later has this capability).

According to the above analysis, our model is closest to the Type III model in [24], and the key difference is that $\mathcal{A}$ in Type III model is not provided with the capabilities C-3 and C-4. Hence, Type III may fail to deal with some important security features, such as forward secrecy and resistance to known key attack. As compared to Li-Lee's model [45] and Yang et al.'s model [22], our model has explicitly taken the malicious card reader into consideration, and $\mathcal{A}$ is further armed with C-3 and C-4.

Moreover, $\mathcal{A}$ in our model is assumed to be able to offline enumerate all the (ID, PW) pairs in the Cartesian product $\mathcal{D}_{id} \times \mathcal{D}_{pw}$ within polynomial time, which enables our model to deal with the special security issues such as resistance to offline password (more precisely, (ID, PW) pair) guessing attack and undetectable online password guessing attack, in dynamic-ID-based schemes (e.g., [41], [44]). Note that, C-02 has also been yet implicitly made in [22], [24], both of which do not concern the admired feature of user anonymity, for the emphasis on C-02 (e.g., we deliberately separate it from C-01 and list it as an independent item) is meaningful only when user anonymity is considered. All in all, our model is stronger and practically reasonable as it incorporates the previous assumptions as well as other new practical assumptions (i.e., the computational power of $\mathcal{A}$ is large but not omnipotent), especially when considering the proliferation of mobile device use cases. Particularly, the practicality of our model is confirmed by the fact that, under such a strong model, secure and efficient protocols can still be built (see Section 4).

2.2 Evaluation Criteria

As pointed out by Yang et al. [22], although the construction and security analysis of smart-card-based password authentication schemes have a long history, there is no common set of desirable security properties that has been widely adopted for the construction of this type of schemes. Later on, Madhusudhan and Mittal [25] showed that earlier criteria sets have redundancies and ambiguities, and hence they proposed a new criteria set of nine security goals and ten desirable features. Since the security goals of their criteria are based on the non-tamper resistance assumption of smart cards, their set is superior to other proposed sets. However, it still has some redundancies (as will be discussed later) and also fails to notice some inherent conflicts (see [21]) among the criteria.

Considering these earlier criteria-related studies [21], [22], [25], based on our cryptanalysis experience of 67 typical schemes (see Appendix A in the supplemental file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2016.2605087) and some of our earlier attacking results [21], [46], [47], and further using the iteration methodology [1] for criterion refinement, we put forward a broad list of 12 independent criteria in terms of user friendliness and security that a two-factor scheme shall satisfy:

- C1. *No password verifier-table*: The server does not need to maintain a database for storing user passwords or some derived values of user passwords;
- C2. *Password friendly*: The password is memorable, and can be chosen freely and changed locally by the user;
- C3. *No password exposure*: The password cannot be derived by the privileged administrator of the server;
- C4. *No smart card loss attack*: The scheme is free from smart card loss attack, i.e., unauthorized users getting a victim's card should not be able to easily change the password of the smart card, recover the victim's password by using online, offline or hybrid guessing attacks, or impersonate the user to login to the system, even if the smart card is obtained and/or secret data in the smart card is revealed;
- C5. *Resistance to known attacks*: The scheme resists various kinds of basic/sophisticated attacks, including offline password guessing attack, replay attack, parallel session attack, de-synchronization attack, stolen verifier attack, impersonation attack, key control, unknown key share attack and known key attack;
- C6. *Sound repairability*: The scheme provides smart card revocation with good repairability, i.e., a user can revoke her card without changing her identity;
- C7. *Provision of key agreement*: The client and the server can establish a common session key for secure data communications during the authentication process;
- C8. *No clock synchronization*: The scheme is not prone to the problems of clock synchronization and time-delay, i.e., the server needs not to synchronize its time clock with these time clocks of all input devices used by smart cards, and vice versa;
- C9. *Timely typo detection*: The user will be timely notified if she inputs a wrong password by mistake when login;
- C10. *Mutual authentication*: The user and server can verify the authenticity of each other.
- C11. *User anonymity*: The scheme can protect user identity and prevent user activities from being traced;
- C12. *Forward secrecy*: The scheme provides the property of perfect forward secrecy.

It is worth pointing out that the criterion C4 deals with attacking scenarios where $\mathcal{A}$ has obtained access to the victim's smart card, while C5 deals with scenarios where $\mathcal{A}$

has *no* access to the victim's smart card. The criterion C4 considers the traditional smart-card-loss issues (see [25]) as well as attacking scenarios newly revealed (e.g., attacks by interactively using the server as an oracle [23] and attacks by returning the extracted card [21]), and see [47] for a taxonomy of eight attacking scenarios. C5 is based on the list of basic attacks [35], [36] that a password-only authentication scheme needs to guard against and on security notions [48] that relate to session keys,[2] as well as on new attack vectors (e.g., stolen verifier attack) that arise in the two-factor authentication environment.

We now show that our criteria set not only eliminates the redundancies and ambiguities of the conventional criteria sets, but also facilitates cryptanalysis due to its concreteness. We first take Madhusudhan and Mittal's set [25], the most recent and representative set ever proposed, as a concrete example of redundancies. Its criteria "SR9. Insider attack" and "G3. No password reveal:" essentially mean the same thing, while its criterion "G4. Password dependent" is completely included in its criterion "SR6. Smart card loss attack", because a scheme which is not password dependent will be prone to smart card loss attack but not to other attack (s). Besides, the important property "free password change", which is widely considered in other sets like [22], [26], is missing in [25]. One can check that the criteria in [25] is entirely included into our set.

Then, we proceed to show the ambiguities of the previous requirement sets. Unlike the criteria set proposed in Liao et al. [49], the criterion concerning with performance, which says "The scheme must be efficient and practical", is not incorporated into our set. The main reason is that, this criterion does not seem to be measurable (and thus ambiguous) without referring to other related schemes. In other words, isolating it from the criteria set can make our set more concrete and decidable. Further, the efficiency of a scheme may depend on the implementation environment, while practicality is largely relevant to the target applications. Except this criterion, all the other criteria in [26], [49] are included into our set. Although the criterion related with performance is not listed in Yang et al.'s set [22], their set is merely composed of four criteria (i.e., C2$\sim$C5) and evidently too limited to be of practical operability.

Summary. Extensive comparisons show that, our adversary model is the harshest one so far and our criterion set is more concrete, concise, and comprehensive as compared to existing works. As any cryptographic protocol meets its goals only within some security model [21], [35], we expect it is the systematic evaluation framework, *as a whole*, that constitutes the main long-term scientific value, but neither our adversary model nor our criteria set alone does. The effectiveness of this framework is tested by rating 67 two-factor schemes without hidden agenda, as summarized in a carefully constructed table in Appendix A, available in the online supplemental material. Both the rating criteria and their definitions were iteratively refined and re-categorized when evaluating these 67 schemes. Each criterion can be *satisfied* by at least 15 of these 67 schemes and in the meantime, it is also *unmet* by at least 7 schemes. This suggests the *necessity* of each criterion. On the other hand, each scheme fails to fulfill at least one criterion, which implies the comprehensiveness of our list and highlights the need for more efforts to design better schemes.

3 Formal Security Model

To formally capture the capabilities of an adversary in smart-card-based password authentication and specify how the adversary interacts with honest parties, we recall the BPR2000 security model [35] where the adversary's capabilities are modelled through queries and define some security notions. However, we do not use the original BPR2000 model directly, but adopt the reified version proposed by Bresson et al. [50] with a few key modifications so that we can define the special security goals (e.g., security against smart card loss attack) for two-factor authentication. We refer the reader unfamiliar with the BPR2000 model to [35], [50] for more details.

Players. In a two-factor protocol $\mathcal{P}$, there are two protocol participants involved, namely, a user $U \in \mathsf{User}$ and a server $S \in \mathsf{Server}$, where User and Server are disjoint. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of $\mathcal{P}$. We denote client instances and server instances by U^i and $S^j, i, j \in \mathbb{Z}$, and denote any kind of instance by $I \in \mathsf{User} \cup \mathsf{Server}$.

Long-Lived Keys. In the registration phase, long-lived keys and public parameters (if any) are established for each participant. The server S is provided with a pair of long-term public and private keys (pub_{key}, pri_{key}) in a public-key based scheme (or a single symmetric key sym_{key} in a symmetric-key based scheme), while each user $U \in \mathsf{User}$ with identity ID_U is equipped with a password PW_U which is assumed to be drawn from a Zipf-distributed [6] "dictionary" $\mathcal{D}$ of small size $|\mathcal{D}|$, where $|\mathcal{D}|$ is a fixed constant which is independent of the system security parameter. The victor $< ID_U, TPW_U >_{U \in \mathrm{User}}$ is kept on S, where TPW_U is an injective transformation of $< ID_U, PW_U >$ and pri_{key}/sym_{key}. Additionally, S stores some non-sensitive user-specific data as well as a few necessary public parameters into a smart card and issues it to U.

Queries. The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. The query types available to $\mathcal{A}$ are defined as follows.

- Execute(U^i, S^j): This oracle query is used to model passive (eavesdropping) attacks of the adversary. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.
- Send(I, m): This query models an active attack, in which $\mathcal{A}$ may send a message to instance I and get back the response that I generates in processing the message m according to the protocol $\mathcal{P}$. A query Send$(U^i$, Start) initializes the protocol. Start is a message, and thus $\mathcal{A}$ receives the flow that the client should send out to S.
- Test(I): This oracle query is not used to simulate the adversary's attack, but to define session key's

2. We do not consider the key compromise impersonation (KCI) attack, because authentication protocols in which the server also serves as the registration center of clients are inherently unable to resist KCI attacks. For more details, readers are referred to [35].

semantic security. If no session key for instance I is defined, then undefined symbol $\perp$ is returned. Otherwise, a private coin c is flipped. If $c = 1$ then the session key sk is returned to $\mathcal{A}$, otherwise a random key of the same size is returned. This query can only be directed towards a fresh instance and called only once during its execution.

- Reveal(I): This query models the misuse of session keys (see C-3 in Table 1). It returns the session key sk of participant instance I to the adversary, if the target instance actually "holds" a session key, and I and its partner were not asked by a Test query. Otherwise the $\perp$ is returned.
- Corrupt(I, a): This query models corruption capability of $\mathcal{A}$ (see C-2 and C-4 in Table 1). $\mathcal{A}$ can break either one of U's two authentication factors, but not both:
  - If $I = U, a = 1$, it outputs the password PW_U of U; If $I = U, a = 2$, it outputs all the security parameters that are stored in the smart card.
  - If $I = S, a = 1$, it outputs S's private key pri_{key} (or the symmetric key pri_{key}) and $< ID_U, TPW_U >_{U \in \text{User}}$ that are stored in S's backend database.

It is not difficult to check that, the above oracle queries indeed can characterize all the adversary's capabilities as specified in Section 2.1, and thus our model defined here facilitates capturing various known attacks, such as impersonation, smart card loss, stolen-verifier, offline password guessing, known key, forward secrecy and passive eavesdropping. Here we mainly focus on security notions and as for the formal definition of user anonymity, readers are referred to a follow-up study [32].

Partnering. We define *partnering* by using the notion of session identifier sid. This notion is essential in the later formulation of *Freshness*. Let U^i and S^j be a pair of instances. We say that the instances U^i and S^j are partnered if the following conditions are true: ① Both instances have accepted; ② Both instances shared the same sid; ③ The partner identifier (pid) of U^i is S and vice-versa. In general, as with [3], we let sid be the ordered concatenation of all messages sent and received by the instance U^i (or S^j).

Freshness. The *freshness* notion is a key point in the definition of protocol security and captures the intuitive fact that a session key can not be trivially known to $\mathcal{A}$. We say that an instance I is fresh if: ① I has accepted and computed a session key; ② Neither I nor its partner has been asked for a Reveal-query; ③ At most one kind of Corrupt-query is made to U from the beginning of the game.

Correctness. If U^i and S^j are partnered and accepted, then they end up with the same session key $sk_U^i = sk_S^j$.

Authentication. A fundamental goal of the authentication schemes is to prevent $\mathcal{A}$ from impersonating the client or the server. We denote by $\mathsf{Adv}_{\mathcal{P}}^{auth}(\mathcal{A})$ the probability that $\mathcal{A}$ successfully impersonates as an instance of either U or S in an execution of $\mathcal{P}$. This means that S (resp. U) agrees on a key, while it is shared with no instance of U (resp. S).

Now it remains to define what is a secure two-factor authentication protocol. For simplicity, here we do not take into account forward secrecy. According to the security results reported on password-only protocols [3], [36], the

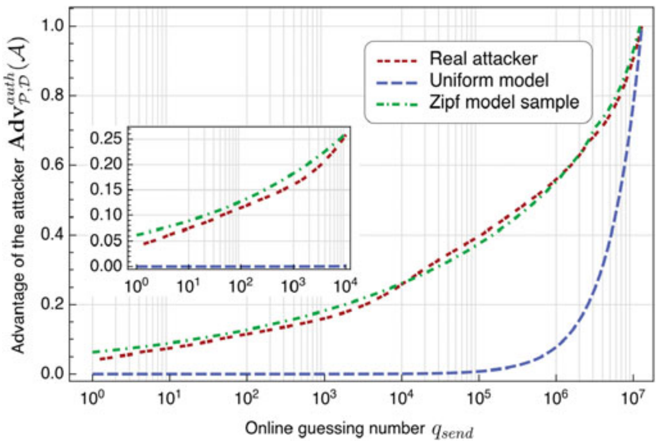

Fig. 3. Online guessing advantages of a real attacker, the uniform-modeled attacker and our Zipf-modeled attacker (using 30.9 million Tianya passwords). Our Zipf attacker well approximates the real attacker.

best strategy that $\mathcal{A}$ in a *secure* password-only protocol can adopt is to perform the online password guessing attack (and test one password candidate in each guessing session), which can be relatively easily detected and conquered.

As for two-factor schemes based on the (conditional) non-tamper resistance assumption of smart cards, it is desirable that online password guessing attack shall also be $\mathcal{A}$'s best possible strategy to impersonate a party: (1) if the password is compromised but not the smart card, $\mathcal{A}$ can hardly succeed because there are still high-entropy secrets stored on the card; and (2) if the smart card is compromised but not the password, $\mathcal{A}$ can at least try one password candidate each time to impersonate the victim by interacting with the server, which is essentially the online guessing attack. Note that, instances with which the adversary interacts via Execute-queries are not counted as on-line attacks. This motivates the following definition: The two-factor protocol $\mathcal{P}$ is said to achieve mutual authentication if, for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ making at most q_{send} on-line attacks, there exists a negligible function $\epsilon(\cdot)$ such that

$$\mathsf{Adv}_{\mathcal{P},\mathcal{D}}^{auth}(\mathcal{A}) \leq C' \cdot q_{send}^{s'} + \epsilon(\ell),$$

where $\mathcal{D}$ is a password space with its frequency distribution follows a Zipf's law [6], C' and s' are the Zipf parameters, and ℓ is the system security parameter. Here we use the Zipf model of Tianya, where $|\mathcal{D}|$=12,898,437, C'=0.062239 and s'=0.155478 [6]. This result requires that calls to the Execute-query (i.e., passive attacks) shall be useless to $\mathcal{A}$.

To help gain a better understanding of the soundness of our above Zipf-model-based formulation (i.e., $C' \cdot q_{send}^{s'} + \epsilon$) over the existing uniform-model-based formulation (i.e., $q_{send}/|\mathcal{D}| + \epsilon$, and readers are referred to some concrete schemes in [22], [26], [30], [31]), we take the Zipf model of Tianya in [6] as an example. Fig. 3 shows the gaps among the *captured* advantages of the real attacker, the uniform-modelled attacker and our Zipf-modelled attacker. As $|\mathcal{D}|$ is generally assumed to be in millions and q_{send} to be in thousands [33], the advantage of a uniform-modelled attacker (i.e., $q_{send}/|\mathcal{D}|$) will be *theoretically* less than 1.0 percent. However, this value for *the real attacker* is about 25.78 percent. Fortunately, our Zipf attacker well approximates the real attacker, approximating $\mathcal{A}$'s advantage to be 26.06

TABLE 2
Notations and Abbreviations

Symbol	Description	Symbol	Description
U_i	ith user	x	the secret key of S
S	remote server	$\oplus$	the bitwise XOR operation
$\mathcal{A}$	the adversary	$\|$	string concatenation
ID_i	identity of user U_i	$\Rightarrow$	a secure channel
PW_i	password of user U_i	$\rightarrow$	a common channel

percent. Fig. 3 shows the uniform-model always greatly underestimates $\mathcal{A}$'s advantage when the guess number q_{send} is below 10^7. In contrast, the Zipf-model slightly overestimates $\mathcal{A}$'s real advantage when $q_{send} \leq 10^4$; when q_{send} is over 10^4 but below 10^6, our Zipf-model slightly underestimates $\mathcal{A}$'s real advantage. This demonstrates that *our Zipf-modelled formulation is particularly suitable for capturing $\mathcal{A}$'s online guessing advantage* (i.e., when the guess number is small, e.g., $q_{send} \leq 10^6$).

Semantic Security. Another major concern of authentication schemes with key agreement is to protect the privacy of the session key. In a protocol run of $\mathcal{P}$, $\mathcal{A}$ can ask a polynomial number of Execute-query, Reveal-query and Send-query. It can also ask a single Test query to a *fresh* instance. At the end of the game, $\mathcal{A}$ outputs a guess bit c' for the bit c involved in the Test-query. We say that $\mathcal{A}$ wins the game if $c' = c$, and this event is denoted by *Succ*. Accordingly, the advantage of $\mathcal{A}$ in breaking the semantic security of the protocol $\mathcal{P}$ is defined as

$$\mathsf{Adv}_{\mathcal{P}}^{ake}(\mathcal{A}) = 2\Pr[\mathbf{Succ}(\mathcal{A})] - 1 = 2\Pr[c' = c] - 1,$$

where the probability space is over all the random coins of the adversary and all the oracles.

The discussions about the notion of "Authentication" also motivate the following definition: The two-factor protocol $\mathcal{P}$ is said to be semantically secure if, for any probabilistic polynomial time adversary $\mathcal{A}$ making at most q_{send} online attacks, there exists a negligible function $\epsilon(\cdot)$ such that

$$\mathsf{Adv}_{\mathcal{P},\mathcal{D}}^{ake}(\mathcal{A}) \leq C' \cdot q_{send}^{s'} + \epsilon(\ell),$$

where the parameters are the same with those of the definition of "Authentication". This result requires that calls to the Execute-query are useless to $\mathcal{A}$.

4 Our Proposed Scheme

In this section, we present a simple, robust yet efficient smart-card-based password authentication scheme that is able to provide all of the twelve criteria introduced in Section 2.2. Our scheme consists of four phases: registration, login, verification and password-change. For ease of presentation, we employ some intuitive abbreviations and notations listed in Table 2.

4.1 Registration Phase

The protocol is defined over a finite cyclic group $\mathbb{G} = < g >$ of order a ℓ-bit prime number q. This group could be a finite group, or it could be an elliptic curve group. In this paper, we assume $\mathbb{G}$ is a prime order subgroup of $\mathbb{Z}_p^*$, where $\mathbb{Z}_p^* = 1, 2, \ldots p-1$ and p also is a large prime number such that $q|p-1$. Hash functions from $\{0,1\}^* \rightarrow \{0,1\}^{l_i}$ are denoted by $\mathcal{H}_i(\cdot)$, where l_i is the bit length of function output (e.g., $l_i = 160$) and $i = 0,1,2,3$. We also define a medium integer n_0, $2^4 \leq n_0 \leq 2^8$, which determines the capacity of the pool of the (ID, PW) pair against online guessing (and relates, as we will show later, to the fuzzy-verifier A_i). Let $(x, y = g^x \bmod p)$ denote the server S's private key and its corresponding public key, where x is kept secret by S and y is stored inside each user's smart card. The registration phase performs as follows:

Step R1. U_i chooses her identity ID_i, password PW_i and a random string b.

Step R2. $U_i \Rightarrow S : \{ID_i, \mathcal{H}_0(b\|PW_i)\}$.

Step R3. On receiving the registration message from U_i at time T, S first picks a random number a_i and computes $A_i = \mathcal{H}_0((\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0(b\|PW_i)) \bmod n_0)$. Then S checks whether U_i is a registered user. If it is U_i's initial registration, S creates an *new* entry for U_i in the account-database and stores $\{ID_i, T_{reg}{=}T, a_i, \texttt{Honey\_List} = \textsc{Null}\}$ in this entry. Otherwise, S only updates the values of T_{reg} to T, a_i to newly created a_i, and $\texttt{Honey\_List}$ to $\textsc{Null}$ in the *existing* entry for U_i. Next, S computes $N_i = \mathcal{H}_0(b\|PW_i) \oplus \mathcal{H}_0(x\|ID_i\|T_{reg})$.

Step R4. $S \Rightarrow U_i$: A smart card with security parameters $\{N_i, A_i, A_i \oplus a_i, q, g, y, n_0, \mathcal{H}_0(\cdot), \ldots, \mathcal{H}_3(\cdot)\}$.

Step R5. Upon receiving the smart card SC, U_i actives it. Then SC requests U_i to enter the random string b twice to confirm its correctness.

Note that, in Step R1 the user U_i may write the random string b on a piece of paper when choosing b, and this paper can be torn once b has been entered into the smart card after the completion of Step R5. In this way, there is no need for U_i to remember b at any time, and thus b can be selected by U_i as random (long and unpredictable) as possible to attain C3: no password exposure (see Section 2.2). In addition, we assume, for simplicity, that the record T_{reg} is secure enough (e.g., 128 bits) against brute-force guessing. In some contexts, T may be only 64-bits long or even shorter, in this case we can set $T_{reg} = T \parallel X$, where X is a large random number.

4.2 Login Phase

This phase involves the following operations:

Step L1. U_i inserts her smart card SC into the card reader and inputs ID_i^*, PW_i^*.

Step L2. SC computes $A_i^* = \mathcal{H}_0((\mathcal{H}_0(ID_i^*) \oplus \mathcal{H}_0(b\|PW_i^*)) \bmod n_0)$ and verifies the validity of ID_i^* and PW_i^* by checking whether A_i^* equals the stored A_i. If they are not equal, the session is terminated.

Step L3. SC chooses a random number u and computes $C_1 = g^u \bmod p, Y_1 = y^u \bmod p, k = \mathcal{H}_0(x\|ID_i\|T_{reg}) = N_i \oplus \mathcal{H}_0(b\|PW_i^*), a_i = (A_i \oplus a_i) \oplus A_i, CID_i = ID_i^* \oplus \mathcal{H}_0(C_1\|Y_1), CAK_i = (a_i\|k) \oplus \mathcal{H}_0(Y_1\|C_1)$ and $M_i = \mathcal{H}_0(Y_1\|k\|CID_i\|CAK_i)$.

Step L4. $U_i \rightarrow S : \{C_1, CID_i, CAK_i, M_i\}$.

Note that, the input to $\mathcal{H}_0(\cdot)$ in the formulation of CAK_i is reversed as compared to that of $\mathcal{H}_0(\cdot)$ in CID_i, in order to prevent information leakage. Otherwise, $\mathcal{A}$ can recover $a_i\|k$

by computing $CID_i \oplus CAK_i$. In addition, the verifier M_i in the login request is added to cope with $\mathcal{A}$'s ability C–02, i.e., to resist against a new kind of smart-card loss attack uncovered by Huang et al. in 2014 [23].

4.3 Verification Phase

After receiving the login request $\{C_1, CID_i, CAK_i, M_i\}$, the server S performs the following operations:

Step V1. S computes $Y_1 = (C_1)^x \bmod p$ using its private key x. Then, S derives $ID_i = CID_i \oplus \mathcal{H}_0(C_1 \| Y_1)$ and checks whether ID_i is in the correct format. If ID_i is not valid, the session is terminated. Otherwise, S proceeds to the next step.

Step V2. S computes k=$\mathcal{H}_0(x \| ID_i \| T_{reg})$ and M_i^*=$\mathcal{H}_0(Y_1 \| k \| CID_i \| CAK_i)$, where T_{reg} is extracted from the entry corresponding to ID_i in the account database. If $M_i^* \neq M_i$, the session is terminated.

Step V3. S derives $a_i' \| k' = CAK_i \oplus \mathcal{H}_0(Y_1 \| C_1)$, and checks whether a_i' equals the stored a_i. An equality implies a login request with the correct A_i. S rejects if they are unequal. Then, S checks whether the derived k' equals the computed k. If they are equal, S proceeds to the next step. If they are unequal, S now knows that $a_i' = a_i$ but $k' \neq k$, implying that there is a $1 - \frac{1}{2^{n_0}}$ probability that U_i's card has been corrupted. Accordingly, S performs either (1) inserts k' into `Honey_List` when there are less than m_0 (e.g., $m_0 = 10$) items in `Honey_List`; or (2) suspends U_i's card (i.e., when there are m_0 items in `Honey_List`) until U_i re-registers.

Step V4. S generates a random number v and computes the temporary key $K_S = (C_1)^v \bmod p$, $C_2 = g^v \bmod p$ and $C_3 = \mathcal{H}_1(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_S)$.

Step V5. $S \rightarrow U_i : \{C_2, C_3\}$.

Step V6. On receiving the reply message from the server S, the smart card computes $K_U = (C_2)^u \bmod p$, $C_3^* = \mathcal{H}_1(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_U)$, and compares C_3^* with the received C_3. This equivalency authenticates the legitimacy of the server S, and U_i goes on to compute $C_4 = \mathcal{H}_2(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_U)$.

Step V7. $U_i \rightarrow S : \{C_4\}$

Step V8. Upon receiving $\{C_4\}$ from U_i, S first computes $C_4^* = \mathcal{H}_2(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_S)$ and then checks if C_4^* equals the received C_4. If this verification holds, S authenticates U_i and the login request is accepted. Otherwise, the connection is terminated.

Step V9. The user U_i and the server S agree on the common session key $sk_U = \mathcal{H}_3(ID_i \| ID_S \| Y_1 \ \| C_2 \| k \| K_U) = \mathcal{H}_3(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_S) = sk_S$ for securing future data communications.

4.4 Password Change Phase

For the sake of security, user friendliness and communication efficiency (i.e., to satisfy the criterion C2), this phase is performed locally without the hassle of interaction with the remote server, and it involves the following steps:

Step P1. U_i inserts her smart card into the card reader and inputs ID_i and the original password PW_i.

Step P2. The card computes $A_i^* = \mathcal{H}_0((\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0(b \| PW_i)) \bmod n_0)$ and verifies the validity of A_i^* by checking whether A_i^* equals to the stored A_i. If the verification holds, it implies the input ID_i and PW_i are valid with a probability of $\frac{n_0-1}{n_0} (\approx \frac{99.61}{100}$, when n_0=2^8). Otherwise, the smart card rejects.

Step P3. The smart card asks U_i to resubmit a new password PW_i^{new} and computes $N_i^{new} = N_i \oplus \mathcal{H}_0(b \| PW_i) \oplus \mathcal{H}_0(b \| PW_i^{new})$, $A_i^{new} = \mathcal{H}_0((\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0 (b \| PW_i^{new}) \bmod n_0)$. Then, smart card updates the values of N_i, A_i and $a_i \oplus A_i$ with N_i^{new}, A_i^{new} and $a_i \oplus A_i^{new}$, respectively.

5 Protocol Design Rationales

In this section, we sketch the basic design ideas behind our protocol and show that "two birds" are hit with one stone: the integration of "honeywords" with a "fuzzy-verifier" not only *eliminates the long-standing security-usability tension* but also *achieves security beyond the conventional optimal bound*.

5.1 Basic Ideas

To achieve the most essential goal—"truly two-factor security" [47], a password-protected cryptographically strong long-term secret $k = \mathcal{H}_0(x \| ID_i \| T_{reg})$ is kept on the smart card, where x is the server S's secret key. On the one hand, this long-term secret k can be derived by S if it knows U_i's identity ID_i and the time of U_i's registration T_{reg}. To this end, a table $\{ID_i, T_{reg}\}$ of registered users is maintained by S. This, in the mean time, preserves C1. On the other hand, k is effectively protected by the password so that breaching the smart card security still does not disclose it. However, U_i herself can reveal it by computing $k = N_i \oplus \mathcal{H}_0(b \| PW_i^*)$, where N_i and b is stored on the card.

To achieve "local and secure password update" (i.e., C2) and address the smart card loss problem (i.e., C4) at the same time, a verification of the authenticity of the original password before updating the value of N_i in the smart card is essential. And thus, besides N_i, some additional parameter(s) should be stored in the card memory, which may introduce new vulnerabilities, such as offline password guessing and user impersonation. To gain a better insight into the subtleties, now let's assume an additional parameter $A_i = \mathcal{H}_0(ID_i \| \mathcal{H}_0(PW_i))$ is stored in the card. Whenever U_i wants to change her password, first she must submit her identity ID_i^* and password PW_i^*, then the card checks if $\mathcal{H}_0(ID_i^* \| \mathcal{H}_0(PW_i^*))$ equals the stored A_i. One can easily find that $\mathcal{A}$ can exhaustively search the correct (ID_i, PW_i) pair in an offline manner once A_i is extracted, which definitely leads to an offline guessing attack, resulting in the violation of C4. What we have just described directly applies to the schemes in [41], [44], [51], [52], where the parameter A_i is exactly computed in this insecure manner and thus $\mathcal{A}$ can obtain the correct (ID_i, PW_i) pair once A_i has been revealed under the capability C-2(ii) in Table 1.

However, if the parameter A_i is computed as $A_i = \mathcal{H}_0 (\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0(PW_i)) \bmod n_0)$, one can be assured that there exists $\frac{|\mathcal{D}_{id}| * |\mathcal{D}|}{n_0} \approx 2^{32}$ candidates of (ID, PW) pair to frustrate $\mathcal{A}$ when $|\mathcal{D}_{id}| = |\mathcal{D}| = 10^6$ [7], [33] and $n_0 = 2^8$, where $|\mathcal{D}_{id}|$

and $|\mathcal{D}|$ denote the size of the identity space and password space, respectively. Even with the capability of C-02 (i.e., the victim user's identity has already been learnt), $\mathcal{A}$ will still be frustrated, because there exist $\frac{|\mathcal{D}|}{n_0} \approx 2^{12}$ password candidates (as will be empirically established in Section 5.2). To further exclude the specious passwords from the remaining 2^{12} candidates, there is no other way than launching an online password guessing attack by interacting with S to determine the exactly correct one. This can be effectively prevented by our introduction of "honeywords" [34] (i.e., a Honey_List in this work) into protocol design to *timely* detect the event that the parameters in U_i's card have been extracted. In Section 5.3, we will demonstrate that this event can be timely detected with an accuracy of $1 - \frac{1}{2^{80}}$; In Section 6, we will rigorously show that this is the best strategy that $\mathcal{A}$ can exploit to obtain the password and thereby to break the protocol. In this manner, we thwart $\mathcal{A}$ from obtaining the correct (ID, PW) pair and we call the parameter A_i calculated through this new method "a fuzzy verifier".

Note that, we do not directly store A_i on the server side as in [34], but instead store a random number a_i corresponding to A_i on S and also store both A_i and $a_i \oplus A_i$ on the smart card, in order to eliminate the risks when S is compromised and A_i is leaked (i.e., the stolen-verifier attack). An obvious "side effect" of this "fuzzy verifier" is to achieve timely typo detection (C9). A scheme with C9 ensures that, in case U_i accidently keys a wrong (ID_i^*, PW_i^*) pair, this event can be timely detected, thereby avoiding fruitless time, computation and communication cost and user fatigue.

We also note that an adversary $\mathcal{A}$ who gets *temporary* access to U_i's smart card may exploit this "security-usability trade-off parameter" A_i. More specifically, $\mathcal{A}$ may attempt to change U_i's password to a new one and then: (i) tries to login; or (ii) returns the card back. In the case i, even though $\mathcal{A}$ somehow guesses PW_i to be PW_i^r such that $\mathcal{H}_0((\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0(b \parallel PW_i^r)) \bmod n_0)$ equals the stored A_i, yet there is only a chance of $\frac{1}{2^{12}}$ that PW_i^r equals PW_i (see Section 5.2). *Only* when $PW_i = PW_i^r$, $\mathcal{A}$ (and the smart card) can retrieve the correct $k = \mathcal{H}_0(x \| ID_i \| T_{reg}) = N_i \oplus \mathcal{H}_0(b \| PW_i^r)$ and successfully logins, because a bogus k will be rejected by the server S in Step V2 of the protocol and also be detected by our honeywords (see Section 5.3). The case ii constitutes a denial of service (DoS) attack. If the threshold of consecutive password change failures is set to five per day (By convention and practice, it is assumed that counter protection of the smart card is in place) and $\mathcal{H}_0(\cdots)$ outputs randomly (which is widely assumed [35]), $\mathcal{A}$ will succeed with a probability only about $\frac{1.95}{100} (\approx \frac{5}{n_0} = \frac{5}{2^8})$. This means $\mathcal{A}$ will not succeed easily (i.e., with a chance of 50 percent after 26 days of attack). Even if this DoS attack succeeds, U_i can restore her card by re-registration.

As discussed above and will be proved in Section 6, this DoS attack and the aforementioned online guessing attack are the two greatest threats that $\mathcal{A}$ poses to our protocol. Fortunately, $\mathcal{A}$ benefits little from this DoS attack and $\mathcal{A}$'s incentive shall be low, while the latter attack can be timely detected and our scheme achieves security beyond the conventional optimal security bound (i.e., beyond $q_{send}/|\mathcal{D}| + \epsilon$).

To avoid password exposure (C3), $\mathcal{H}_0(b \| PW_i)$ instead of PW_i or $h(PW_i)$ is submitted to the server S, where b is a random number unknown to the server S; to achieve C6, an entry (ID_i, T_{reg}) corresponding to U_i is stored in S's database, only T_{reg} needs to be updated when user U_i revokes her smart card; to avoid clock synchronization (C8), a nonce based mechanism instead of the timestamp based design is preferred to provide the freshness of the messages; to achieve user anonymity (C11), user's real identity ID_i is concealed in the session-variant pseudo-identity CID_i, the formal proof for C11 is essentially the same with the Theorem 2 in [32] which is based on the decisional Diffie-Hellman assumption; to achieve forward secrecy (C12), DH key exchange technique is adopted; and C4 and C10 will be further rigorously proved in Section 6.

5.2 Effectiveness of "Fuzzy-Verifier"

We now use large-scale real-life passwords to show that our proposed "fuzzy-verifier" $A_i = \mathcal{H}_0((\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0(b \parallel PW_i)) \bmod n_0)$ is effective in dividing $\mathcal{A}$'s password guessing space, leaving adequate candidates for $\mathcal{A}$ to identify and thus making it possible for "honeywords", as will be shown in Section 5.3, to bound $\mathcal{A}$'s advantage to a low value.

Assume $\mathcal{A}$ has obtained U_i's card and extracted the parameter A_i. With the knowledge of A_i, $\mathcal{A}$ can reduce the size of her guessing space $\mathcal{D}$ to $\frac{|\mathcal{D}|}{n_0}$. We now show that $\mathcal{D}/n_0$ is *practically* large enough. It is natural for us to approximate $\mathcal{D}$ by using real-life password accounts *with frequency*; As $\mathcal{A}$ is clever, when performing an online guessing, she would always try the most likely password candidate first [33], and this attacking strategy is best captured by the security notion of guessing entropy (GE):

$$G(\mathcal{D}) = \sum_{i=1}^{|\mathcal{D}|} p_i \cdot i,$$

where, without loss of generality, each password pw_i in $\mathcal{D}$ is assumed to be associated with a probability p_i and $p_1 \geq p_2 \geq p_3 \geq \cdots$. Thus, $G(\mathcal{D})$ is just the expected number of guesses required to find the correct password PW_i.

Here we employ four datasets: 32 million Rockyou [11], 30 million Tianya, 16 million Dodonew and 6 million CSDN [53]. The first dataset was hacked in Dec. 2009 from the popular gaming site Rockyou.com, and the later three datasets were hacked in Dec. 2011 from three high-profile web services in China. They all were made publicly available. For illustration, we set $n_0 = 2^8$. To be practical, we do not use the entire password datasets to approximate $\mathcal{A}$'s guessing space $\mathcal{D}$, but use the most vulnerable distributions (i.e., portions with the top popular passwords). The underlying reasons are that: (1) $\mathcal{A}$ cares about cost-effectiveness and would not try these least popular (low-gain) passwords; (2) If these most vulnerable portions of a dataset can assure satisfactory guessing entropy, then these less vulnerable portions would naturally reach the goal.

Table 3 shows that, when the size of the dataset is no less than 3 million, each divided pool indeed can reach a GE$\geq 2^{12}$ (when n_0=2^8). This indicates that, even if the parameter A_i is extracted (or somehow guessed) by $\mathcal{A}$, there are still 2^{12} password candidates that $\mathcal{A}$ has to guess in an online manner, while online guessing can be easily thwarted. This means that our "fuzzy-verifier" is indeed effective for these services

TABLE 3
Guessing Entropy (GE) Distributions of $n_0 = 2^8$ Password Pools

Real-life password distributions	% of password pools with GE$\geq 2^{12} (= 2^{20}/n_0)$
Rockyou_Top1Million	0.00%
Tianya_Top1Million	10.16%
CSDN_Top1Million	10.54%
Dodonew_Top1Million	14.45%
Rockyou_Top2Million	84.77%
Tianya_Top2Million	96.09%
CSDN_Top2Million	97.66%
Dodonew_Top2Million	98.83%
Rockyou_Top$x$Million($x \geq 3$)	99.61%
Tianya_Top$x$Million($x \geq 3$)	100.00%
CSDN_Top$x$Million($x \geq 3$)	100.00%
Dodonew_Top$x$Million($x \geq 3$)	100.00%

For more details, readers can see http://bit.ly/2b7fkup. Each pool stems from the division of password space $\mathcal{D}$ according to our fuzzy-verifier A_i.

with a user-base ≥3 million, and it will also be effective for services with a smaller scale when we adjust the value of n_0. Dodonew is the strongest one among the four datasets in term of GE, while Rockyou is the least strong one. This has useful implications: for passwords created under a similar context (e.g., password creation policy and service type) to Dodonew, their guessing space shall be as large as 2 million to reach a GE$\geq 2^{12}$, while passwords created under a context similar to Rockyou shall be with a space of 3 million.

5.3 Effectiveness of "Fuzzy-Verifier" + "Honeywords"

In our scheme, to provide the admired property of "local and secure password change" [21], U_i stores the "fuzzy-verifier" A_i=$\mathcal{H}_0((\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0(b \| PW_i)) \bmod n_0)$ in its card memory. This also facilitates $\mathcal{A}$ to reduce her password guessing space size from $|\mathcal{D}|$ to $\frac{|\mathcal{D}|}{n_0}$ in an offline manner. In the above section, we have shown that our "fuzzy-verifier" is effective in making $|\mathcal{D}|/n_0$ large enough in practice. One can see that, if S can effectively detect the event, denoted by Ext, that the parameters in U_i's card have been extracted and S timely suspends the corrupted card, $\mathcal{A}$ is still prevented from gaining a large advantage in determining the final correct password PW_i from the $\frac{|\mathcal{D}|}{n_0}$ candidates by using online guessing. Fortunately, in what follows we show that the integration of "honeywords" (i.e., the items in `Honey_List`) with our "fuzzy-verifier" (i.e., A_i) enables our scheme to detect the event Ext with an accuracy of $1 - \frac{1}{2^{80}}$ after the attacker makes just ten online guessing attempts.

In Step V3 of the Verification phase, whenever the server S finds a login request that is with the *right* A_i but with an *erroneous* k, S is with a confidence $1 - \frac{1}{n_0}$ that the event Ext occurs. Furthermore, if we specify that S revokes U_i when such enormous login events occur m_0 times, then S is with a confidence $1 - (\frac{1}{n_0})^{m_0}$ (e.g., which equals $1 - \frac{1}{2^{80}}$ when $m_0 = 10$ and $n_0 = 2^8$) to be assured that the event Ext occurs. The choices for the two tradeoff parameters m_0 and n_0 are mainly constrained by the following requirements:

R1. Pr[Err_change] $= \frac{1}{n_0}$ shall be as small as possible, where Err_change denotes the event that, U_i accidentally types a password PW_i^* when changing password, yet $\mathcal{H}_0((\mathcal{H}_0(ID_i) \oplus \mathcal{H}_0(b \parallel PW_i^*)) \bmod n_0)$ equals A_i and PW_i is unwittingly changed to PW_i^*.

R2. Pr[Err_detect]$= \frac{1}{(n_0)^{m_0}}$ shall be as small as possible, where Err_detect denotes the event that S incorrectly revokes U_i's smart card. That is, S determines that Ext occurs yet actually, U_i's card has not been corrupted.

R3. Pr[Succ_Ext]$= C' \cdot m_0^{s'}$ shall be as small as possible, where Succ_Ext denotes the event that Ext occurs and $\mathcal{A}$ successfully obtains PW_i by interacting with S.

It is critical to observe that: (1) R1 and R2 require m_0 and n_0 to be as *large* as possible, while R3 requires m_0 and n_0 to be as *small* as possible, and thus we need a balance; and (2) Pr[Err_detect] decreases *exponentially* with m_0, while Pr [Succ_Ext] increases *linearly* with m_0, which means we can timely (and accurately) detect the event Ext and, at the meantime, confine $\mathcal{A}$'s guessing advantage to the possible minimum by keeping m_0 small enough (e.g., $m_0 \in [3, 20]$).

In this work, we recommend $n_0 = 2^8$ and $m_0 = 10$. As discussed above, it is acceptable to set $n_0 = 2^8$; Since it is very undesirable to mistakenly suspend a legitimate user's card, Pr[Err_detect] shall be negligible and when $m_0 \geq 10$,

$$\Pr[\text{Err\_detect}] = \frac{1}{(n_0)^{m_0}} \leq \frac{1}{(2^8)^{10}}.$$

This indicates the event Ext can be detected with an accuracy of $1 - \frac{1}{2^{80}}$. On the other hand, when we set $m_0 \leq 10$,

$$\Pr[\text{Succ\_Ext}] = C' \cdot m_0^{s'} \approx \sum_{j=1}^{m_0} p_i \leq \sum_{j=1}^{10} p_i = 7.43\%,$$

where we use the distribution of 30.9 M Tianya passwords [53] as a concrete example: C'=0.062239 and s'=0.155478.

We summarize in Table 4 the results for 11 real-life password distributions with the guess number (i.e., m_0) varying from $1, 10$ to 10^4, as well as varying from $1/10^4$ percent to $1/5$ percent. We suggest setting m_0=10. We emphasize that, the values for m_0 and n_0 can be adjusted to cater for diversified security demands in different systems.

What's most surprising in Table 4 is that, with just a handful of online guessing attempts, $\mathcal{A}$ can obtain a considerable amount of advantages. For instance, the average advantage of $\mathcal{A}$ will be up to 8.21 percent by only performing 100 online guesses (in an optimal order). This is in vast different from the traditional *theoretical* optimal security bound (see [26], [30], [31]): $q_{send}/|\mathcal{D}| + \epsilon$, where $\mathcal{D}$ is assumed to be uniformly distributed, and q_{send} denotes the number of online guessing that $\mathcal{A}$ engages in. More specifically, since $\mathcal{D}$ is generally assumed to be 10^6 and q_{send} to be in thousands [33], $q_{send}/|\mathcal{D}|$ will be theoretically over 0.1 percent. Yet, the actual value is over 15.09 percent, which is far from a negligible risk level. From the last two rows in Table 4, one can see that there are *two to four orders of magnitude difference* in $\mathcal{A}$'s online guessing advantages between the uniform model and the realistic model (which can be well captured by our Zipf model, see Fig. 3). If we had considered *targeted online guessing* [54] and taken into account the fact that user passwords become weaker when additional authentication factor are in place [55], this gap will be even larger.

TABLE 4
The Cumulative Percentages of Top-x Most Popular Passwords (PWs) of Each Real-Life Password Dataset

Datasets (leaked year)	Language	Total PWs	Unique PWs	Top 1	Top 10	Top 10^2	Top 10^3	Top 10^4	Top $\frac{1}{5}$	Top $\frac{1}{10}$	Top $\frac{1}{10^2}$	Top $\frac{1}{10^3}$	Top $\frac{1}{10^4}$
Rockyou (2009)	English	32,581,870	14,326,970	0.89%	2.05%	4.55%	11.30%	22.31%	64.82%	57.28%	39.30%	24.24%	12.84%
Phpbb (2009)	English	255,373	184,341	1.04%	2.79%	5.70%	12.89%	27.44%	42.25%	34.05%	15.95%	7.06%	3.42%
Singles.org (2010)	English	16,248	12,233	1.36%	3.40%	9.19%	26.35%	86.26%	39.76%	29.09%	10.06%	3.65%	1.36%
Faithwriters (2009)	English	9,708	8,346	0.55%	2.17%	7.76%	24.33%	100.00%	31.22%	22.62%	7.06%	1.90%	0.00%
Tianya (2011)	Chinese	30,901,241	12,898,437	3.98%	7.43%	11.50%	16.04%	25.78%	66.55%	58.21%	41.19%	27.45%	16.70%
Dodonew (2011)	Chinese	16,258,891	10,135,260	1.45%	3.28%	5.60%	8.59%	13.62%	50.13%	39.97%	22.72%	13.66%	8.61%
Csdn (2011)	Chinese	6,428,277	4,037,605	3.66%	10.44%	13.26%	16.54%	23.91%	49.75%	42.66%	28.46%	20.62%	14.97%
Sina weibo (2011)	Chinese	4,730,662	2,828,618	3.74%	7.17%	10.24%	13.96%	21.49%	52.17%	43.82%	27.24%	16.51%	11.71%
Gmail (2014)	hybrid	4,926,650	3,132,028	0.97%	2.07%	3.88%	8.65%	17.76%	49.14%	41.64%	23.78%	12.63%	5.77%
Mail.ru (2014)	Russian	4,932,688	2,949,616	1.82%	4.05%	6.37%	9.94%	17.40%	52.16%	43.88%	24.92%	12.46%	7.81%
Yandex.ru (2014)	Russian	1,261,809	717,202	3.10%	7.66%	12.26%	17.43%	26.53%	54.53%	44.29%	24.53%	16.61%	11.55%
Average above	–	9,300,311	4,657,332	**2.05%**	**4.78%**	**8.21%**	**15.09%**	**28.25%**	50.23%	41.59%	24.11%	14.25%	8.61%
Uniform distribution	–	1,000,000	1,000,000	0.0001%	0.0010%	0.01%	0.10%	1.00%	20.00%	10.00%	1.00%	0.10%	0.01%

All this highlights that, due to highly skewed password distributions in reality, *the conventional optimal security bound largely underestimates $\mathcal{A}$'s advantages and engenders a false sense of security*. It is imperative to prevent $\mathcal{A}$ from trying moderate number of online guessing attempts to bound $\mathcal{A}$'s advantage to a low value. Fortunately, the use of "honeywords" with a "fuzzy-veifier" provides a promising solution to this issue.

In a nutshell, before $\mathcal{A}$ sets off an alarm of user card corruption with a probability $\frac{1}{(n_0)^{m_0}}$ (e.g., $\frac{1}{2^{80}}$) of false positive, she can only mount m_0 ($m_0 \in$ [3,20] as suggested) online guessing attempts, which is her best attacking strategy (for further evidence see Section 6), while in conventional schemes this figure would generally be in hundreds or even in thousands. In this light, our scheme achieves security (i.e., $\Pr[\mathsf{Succ\_Ext}] = C' \cdot m_0^{s'} \approx \sum_{j=1}^{m_0} p_i \leq \sum_{j=1}^{10} p_i$) beyond the conventional optimal security bound (i.e., $q_{send}/|\mathcal{D}| + \epsilon$), serving as a hedge against human-beings' limited memory.

5.4 Wide Applicability of our Approach

We proceed to show that our integration of "honeywords" with "fuzzy-verifiers" is a generic one and can be readily applied to existing two-factor authentication schemes for both single-server architecture [22] and multi-server environment [56]. More specifically, a fuzzy-verifier (like our A_i) is stored on the user's card memory to provide the usability property "local and secure password change" (i.e., C2), while some "honeywords" (like these items in `Honey_List`) are kept on the backend database of the authentication server (or so-called control server in the multi-server environment) to preserve the security goal of "no smart card loss attack" (i.e., C4). Most essentially, "honeywords" enable the system to accurately detect the event that $\mathcal{A}$ is exploiting the fuzzy-verifier as a oracle to reduce the password space and to thwart $\mathcal{A}$'s malicious action in a timely manner (e.g., by rate limiting or locking the account).

In Appendix B, available in the online supplemental material, we use two typical schemes, i.e., Tsai et al.'s scheme [28] and Xue et al.'s scheme [52], as case studies to show exactly how our approach can be integrated into *other two-factor schemes* (for the single-server architecture) and into *two-factor schemes for the multi-server architecture*, respectively. We further employ Odelu et al.'s scheme [56] to briefly show its applicability to three-factor schemes. That is, most of the schemes (e.g., [20], [22], [26], [43]) previously subject to the long-standing C2 versus C4 dilemma now can be relieved by using our proposed approach.

Summary. Both theoretical and empirical results show the practicality of integrating "honeywords" [34] with a "fuzzy-verifier" to well balance the useability feature of "local password update" (C2) and the security goal of "no smart loss problem" (C4). This provides thus far the most promising solution to the open problem left in [19]: "whether or not there exist secure smart-card-based password authentication protocols and the password-changing phase does not need any interaction with the server"?

6 Formal Security Analysis

In the following, we show that our scheme is provably secure in the formal model defined in Section 3, under the assumptions that the hash function closely behaves like a random oracle and that the computational Diffie-Hellman problem is intractable.

We first provide a formal description of the proposed protocol by specifying the registration phase and the oracles to which the adversary has access. Before the registration phase, for security parameter ℓ, the algorithm `Init` first runs an algorithm $\mathcal{G}$ to generate a group $\mathbb{G}$ of prime order q, where $|q| = \ell$. Next, `Init` generates a generator g for $\mathbb{G}$, four collision-resistant hash functions $\mathcal{H}_i : \{0,1\}^* \rightarrow \{0,1\}^{l_i}$ $(i = 0, 1, 2, 3)$, and a long-term private/public key pair $(x, y = g^x)$ for server S. Each user U_i is equipped with a password PW_i which is drawn from a Zipf-distributed dictionary $\mathcal{D}$ of size $|\mathcal{D}|$. Additionally, when the user U_i enrolls in the server S, S stores user-specific secret data $\{N_i, A_i, A_i \oplus a_i\}$ as well as other public parameters into a smart card and issues it to the user U_i, where N_i and A_i are transformations of PW_i and S's private key x. Further, a formal specification of the Execute, Reveal, Corrupt and Test oracles appears in Appendix C, available in the online supplemental material.

Before stating the security results, we recall the computational assumption on which the formal security proof relies.

Computational Diffie–Hellman (CDH) Assumption. Let $\mathbb{G}$ be a finite cyclic group of prime order q generated by an element g, where the operation is denoted multiplicatively.

TABLE 5
Performance Comparison Among Relevant Authentication Schemes

	Protocol rounds[a]	Computation overhead		Communication cost		Storage cost	The proposed twelve evaluation criteria											
		User side	Server side	User side	Server side		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
Xu et al. (2009) [29]	2	$3T_E + 5T_H \approx 481.305$ ms	$3T_E + 4T_H \approx 8.615$ ms	1,408 bits	1,408 bits	3,200 bits	✓	×	×	×[b]	×	×	✓	×	×	✓	×	✓
Wang et al. (2012) [44]	3	$3T_E + 7T_H \approx 505.827$ ms	$3T_E + 5T_H \approx 8.616$ ms	1,408 bits	1,152 bits	3,456 bits	✓	✓	✓	×[c]	✓	×	✓	✓	✓	✓	✓	✓
Wu et al. (2012) [26]	3	$4T_E + 5T_H \approx 621.305$ ms	$3T_E + 5T_H \approx 8.616$ ms	2,304 bits	1,152 bits	3,456 bits	✓	✓	✓	×	✓	✓	✓	✓	✓	×	✓	✓
Li et al. (2013) [51]	2	$4T_E + 4T_H \approx 609.044$ ms	$3T_E + 3T_H \approx 8.615$ ms	1,408 bits	1,408 bits	4,096 bits	✓	✓	×	×[c]	✓	×	✓	×	✓	✓	×	✓
Byun (2015) [31]	2	$5T_E + T_S + 5T_H \approx 766.277$ ms	$5T_E + T_S + 3T_H \approx 14.357$ ms	2,176 bits	1,152 bits	2,176 bits	×	×	✓	✓	✓	✓	✓	✓	×	✓	✓	✓
Jiang et al. (2015) [57]	2	$4T_E + 4T_H \approx 609.044$ ms	$2T_E + 4T_H \approx 5.744$ ms	2,304 bits	1,152 bits	3,328 bits	✓	✓	×	✓	✓	×	✓	×	✓	✓	×	×
Truong et al. (2015) [58]	3	$T_C + 7T_H \approx 525.327$ ms	$3T_C + 7T_H \approx 30.775$ ms	640 bits	1,152 bits	384 bits	✓	✓	×	×[c]	✓	×	✓	✓	✓	✓	×	✓
Islam et al. (2016) [59]	2	$3T_E + 3T_H \approx 456.783$ ms	$2T_E + 3T_H \approx 5.744$ ms	1,408 bits	1,408 bits	2,176 bits	✓	✓	✓	×[c]	✓	✓	✓	✓	✓	✓	✓	✓
Our scheme	3	$3T_E + 9T_H \approx 530.349$ ms	$3T_E + 7T_H \approx 8.617$ ms	1,536 bits	1,152 bits	3,616 bits	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

[a] *All the schemes with two rounds are either prone to clock synchronization issue or reply attack;*
[b] *This scheme is vulnerable to user impersonation as shown in [51];*
[c] *All the schemes in [44], [51], [58], [59] are subject to the smart-card loss problem because they provide "secure and local password change" [21] with explicit verifier.*

A (t, ϵ)-CDH attacker in $\mathbb{G}$ is a PPT machine Δ running in time t such that

$$\mathsf{Adv}^{\mathrm{CDH}}_{g,\mathbb{G}}(\Delta) = \Pr_{x,y}[\Delta(g^x, g^y) = g^{xy}] \geq \epsilon$$
$$\mathsf{Adv}^{\mathrm{CDH}}_{g,\mathbb{G}}(t) = \max_{\Delta}\{\mathsf{Adv}^{\mathrm{CDH}}_{g,\mathbb{G}}(\Delta)\},$$

where the probability is taken over the random values x and y. The CDH-Assumption states that $\mathsf{Adv}^{\mathrm{CDH}}_{g,\mathbb{G}}(t) \leq \epsilon$ for any t/ϵ not too large.

Theorem 1. *Let $\mathbb{G}$ be a representative group, $\mathcal{D}$ be a password space with its frequency distribution following the Zipf's law [6], and n_0 be the "security-usability trade-off parameter". Let $\mathcal{P}$ be the proposed scheme stated in Section 4. Let $\mathcal{A}$ be a PPT adversary against the semantic security within a time bound t, with $q_{send}(\leq m_0 = 10)$ Send-queries and q_{exe} Execution-queries, and making less than q_h random oracle queries. Then we have*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{ake}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) &= 2\Pr[\mathrm{Succ}_7] - 1 + 2(\Pr[\mathrm{Succ}_0] - \Pr[\mathrm{Succ}_7]) \\ &\leq C' \cdot q_{send}^{s'} + 12q_h\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{P}}(t + (q_{send} + q_{exe} \\ &\quad + 1) \cdot \tau_e) + \frac{q_h^2 + 8q_{send}}{2^l} + \frac{(q_{send} + q_{exe})^2}{p},\end{aligned}$$

where we use the Zipf model of the Tianya password distribution in [6], where $|\mathcal{D}|$=12,898,437, C'=0.062239 and s'=0.155478; $n_0 = 2^8$; τ_e is the computation time for an exponentiation in $\mathbb{G}$, and $l = min\{l_i\}, i = 0, 1, 2, 3$.

Proof. Let $\mathcal{A}$ be an adversary against the semantic security of our scheme. Our main idea is to employ $\mathcal{A}$ to construct probabilistic polynomial-time (PPT) adversaries for each of the underlying primitives (e.g., Hash and CDH intractability) in such a way that if $\mathcal{A}$ manages to break the semantic security, then at least one of these PPT adversaries succeeds in breaking the security of an underlying primitive. We prove Theorem 1 through a series of hybrid games $G_n (n = 0, 1, \ldots, 8)$, starting with the real attack G_0 and ending in G_8 where $\mathcal{A}$'s advantage is 0, and for which we can bound the difference in $\mathcal{A}$'s advantage between any two consecutive games. The detailed proof can be found in Appendix C-1, available in the online supplemental material. □

Theorem 2. *$\mathbb{G}$, $\mathcal{D}$, n_0 and $\mathcal{P}$ are of the same meaning with those of Theorem 1. Let $\mathcal{A}$ be an adversary against mutual authentication within a time bound t, with less than $q_{send}(\leq m_0 = 10)$ Send-queries and q_{exe} Execution-queries, and making less than q_h random oracle queries. Then,*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{auth}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) &\leq C' \cdot q_{send}^{s'} + \frac{q_h^2 + 8q_{send}}{2^{l+1}} + \frac{(q_{send} + q_{exe})^2}{2p} \\ &\quad + 5q_h\mathsf{Adv}^{\mathrm{CDH}}_{\mathcal{P}}(t + (q_{send} + q_{exe} + 1) \cdot \tau_e),\end{aligned}$$

where τ_e is the computation time for an exponentiation in $\mathbb{G}$ and $l = min\{l_i\}, i = 0, 1, 2, 3$.

Proof. This proof is similar to that of semantic security. And interested readers are referred to Appendix C-2, available in the online supplemental material. □

7 Performance Evaluation

In this section, we compare the performance and the fulfillment of the criteria among relevant schemes [26], [29], [31], [44], [51], [57], [58], [59] and our proposed scheme. The comparison results are depicted in Table 5. Sixty-seven typical schemes are further evaluated in Appendix A, available in the online supplemental material.

Without loss of generality, the security parameter n_0 is assumed to be 32-bit long, the identity ID_i, password PW_i, random numbers, timestamps and output of hash functions are all recommended to be 128-bit long, while y and g are 1,024-bit long. Let T_H, T_E, T_S, and T_C denote the time complexity for hash, modular exponentiation, symmetric encryption and Chebysev polynomial, respectively. Other lightweight operations like XOR and $\|$ are omitted.

TABLE 6
Timings for Cryptographic Operations

Experimental platform	Exp. T_E ($\|p\| = 1{,}024$)	Chebysev T_C ($\|p\| = 1{,}024$)	Symm. T_S (AES-128)	Hash T_H (SHA-1)
Philips HiPer Smart 36 MHz	140.0 ms	439.5 ms	4.972 ms	12.261 ms
Intel(R) T5870 2.00 GHz	10.257 ms	32.200 ms	2.012 μs	2.580 μs
Intel(R) i7-4790 3.60 GHz	2.871 ms	10.257 ms	0.086 μs	0.598 μs

To have a more intuitive grasp on the computation overhead of our scheme, in Table 6 we list the computation time for related cryptographic operations on different platforms. We use a Philips HiPerSmart card to approximate user device, and the computation time of related operations is reported in [60]. This smart card is equipped with a 32-bit RISC MIPS-based processor, offering a maximum clock speed of 36 MHz, as well as a 2 KB instruction cache, 256 KB flash memory and 16 KB RAM. We use common Laptops to approximate the server and evaluate the server side computation cost. Note that both our implementation and that of [60] make use of the standard cryptographic library MIRACL [61], which is a multi-precision integer and rational arithmetic C/C++ library.

As shown in Table 5, our scheme provides all the twelve criteria while maintaining reasonable efficiency; all the other schemes fail to achieve at least one critical criterion because of security pitfalls or due to a violation of the inherent security-usability conflict revealed in [21]—they only employ conventional cryptographic approaches.

8 Conclusion

In this paper, we have taken a first step towards breaking the “break-fix-break-fix cycle” in the two-factor authentication research area. Beyond our proposal of a new scheme which meets practicability, simplicity, and strong notions of security, the proposed adversary model and criteria set provide a benchmark for the evaluation of current and future two-factor authentication proposals. To the best of our knowledge, we, for the first time, introduce “honeywords” [34], traditionally the purview of system security, into two-factor cryptographic protocol design. By integrating “honeywords” with the proposed “fuzzy-verifier”, our scheme can timely detect user card corruption to thwart online guessing and well addresses the seemingly intractable security-usability issue left in [23]. Particularly, eleven large-scale password datasets, which consist of 102.6 million real-life passwords and cover various popular services and diversified user bases (e.g., language), are used to establish the practicality of the proposed approach.

Acknowledgments

The authors thank the anonymous reviewers for their invaluable comments. All the Appendixes A, B and C can be found in the supplemental file, available in http://doi.ieeecomputersociety.org/10.1109/TDSC.2016.2605087. A preliminary version appeared at https://eprint.iacr.org/2012/439. This research was supported by the National Natural Science Foundation of China under Grant No. 61472016, and by the National Key Research and Development Plan of China under Grant No. 2016YFB0800600.

References

[1] J. Bonneau, C. Herley, P. Oorschot, and F. Stajano, “The quest to replace passwords: A framework for comparative evaluation of Web authentication schemes,” in *Proc. IEEE Symp. Secur. Privacy*, 2012, pp. 553–567.
[2] T. Wu, “The secure remote password protocol,” in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 1998, pp. 1–15.
[3] J. Katz, R. Ostrovsky, and M. Yung, “Efficient and secure authenticated key exchange using weak passwords,” *J. ACM*, vol. 57, no. 1, pp. 1–41, 2009.
[4] M. Abdalla, F. Benhamouda, and P. MacKenzie, “Security of the J-PAKE password-authenticated key exchange protocol,” in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 571–587.
[5] M. Adeptus, “Hashdumps and passwords,” May 2014. [Online]. Available: http://www.adeptus-mechanicus.com/codex/hashpass/hashpass.php
[6] D. Wang and P. Wang, “On the implications of Zipf’s law in passwords,” in *Proc. Eur. Symp. Res. Comput. Secur.*, 2016, pp. 1–21.
[7] J. Ma, W. Yang, M. Luo, and N. Li, “A study of probabilistic password models,” in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 538–552.
[8] J. Gosney, “Password cracking HPC,” in *Proc. Password*, 2012. [Online]. Available: http://bit.ly/1y00I3O
[9] J. Goldberg, “Bcrypt is great, but is password cracking infeasible?” Mar. 2015. [Online]. Available: http://bit.ly/2bpadR8
[10] T. Pham, “Four years later, anthem breached again: Hackers stole credentials,” Feb. 2015. [Online]. Available: http://bit.ly/2btkw6K
[11] C. Allan, “32 million RockYou passwords stolen,” Dec. 2009. [Online]. Available: http://www.hardwareheaven.com/news.php?newsid=526
[12] T. Fox-Brewster, “15 Million passwords appear to have leaked from 000webhost,” Oct. 2015. [Online]. Available: http://www.forbes.com/sites/thomasbrewster/2015/10/28/000webhost-database-leak/
[13] E. Kovacs, “PhpBB asking users to change passwords following hack,” Dec. 2014. [Online]. Available: http://www.securityweek.com/phpbb-asking-users-change-passwords-following-hack
[14] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, “The tangled Web of password reuse,” in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–15.
[15] J. Camenisch, A. Lehmann, and G. Neven, “Optimal distributed password verification,” in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 182–194.
[16] D. Weinshall, “Cognitive authentication schemes safe against spyware,” in *Proc. IEEE Symp. Secur. Privacy*, 2006, pp. 295–230.
[17] Q. Yan, J. Han, Y. Li, and R. H. Deng, “On limitations of designing leakage-resilient password systems: Attacks, principles and usability,” in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2012, pp. 1–16.
[18] C. Chang and T. Wu, “Remote password authentication with smart cards,” *IEE Comput. Digital Tech.*, vol. 138, no. 3, pp. 165–168, May 1991.
[19] X. Huang, Y. Xiang, E. Bertino, J. Zhou, and L. Xu, “Robust multi-factor authentication for fragile communications,” *IEEE Trans. Depend. Secur. Comput.*, vol. 11, no. 6, pp. 568–581, Nov./Dec. 2014.
[20] J. Yu, G. Wang, Y. Mu, and W. Gao, “An efficient generic framework for three-factor authentication with provably secure instantiation,” *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 12, pp. 2302–2313, Dec. 2014.
[21] D. Wang, D. He, P. Wang, and C.-H. Chu, “Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment,” *IEEE Trans. Depend. Secur. Comput.*, vol. 12, no. 4, pp. 428–442, Jul./Aug. 2015.
[22] G. M. Yang, D. S. Wong, H. X. Wang, and X. T. Deng, “Two-factor mutual authentication based on smart cards and passwords,” *J. Comput. Syst. Sci.*, vol. 74, no. 7, pp. 1160–1172, 2008.
[23] X. Huang, X. Chen, J. Li, Y. Xiang, and L. Xu, “Further observations on smart-card-based password-authenticated key agreement in distributed systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1767–1775, Jul. 2014.
[24] Y. G. Wang, “Password protected smart card and memory stick authentication against off-line dictionary attacks,” in *Proc. 27th IFIP TC 11 Inf. Secur. Privacy Conf. Inf. Secur. Privacy Res.*, 2012, pp. 489–500.

[25] R. Madhusudhan and R. Mittal, “Dynamic ID-based remote user password authentication schemes using smart cards: A review,” *J. Netw. Comput. Appl.*, vol. 35, no. 4, pp. 1235–1248, 2012.
[26] S. H. Wu, Y. F. Zhu, and Q. Pu, “Robust smart-cards-based user authentication scheme with user anonymity,” *Secur. Commun. Netw.*, vol. 5, no. 2, pp. 236–248, 2012.
[27] C. Fan, Y. Chan, and Z. Zhang, “Robust remote authentication scheme with smart cards,” *Comput. Secur.*, vol. 24, no. 8, pp. 619–628, 2005.
[28] J.-L. Tsai, N.-W. Lo, and T.-C. Wu, “Novel anonymous authentication scheme using smart cards,” *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2004–2013, Nov. 2013.
[29] J. Xu, W. Zhu, and D. Feng, “An improved smart card based password authentication scheme with provable security,” *Comput. Standards Interfaces*, vol. 31, no. 4, pp. 723–728, 2009.
[30] M. Shirvanian, S. Jarecki, N. Saxena, and N. Nathan, “Two-factor authentication resilient to server compromise using mix-bandwidth devices,” in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–16.
[31] J. W. Byun, “Privacy preserving smartcard-based authentication system with provable security,” *Secur. Commun. Netw.*, vol. 8, no. 17, pp. 3028–3044, 2015.
[32] D. Wang, N. Wang, P. Wang, and S. Qing, “Preserving privacy for free: Efficient and provably secure two-factor authentication scheme with user anonymity,” *Inf. Sci.*, vol. 321, pp. 162–178, 2015.
[33] J. Bonneau, “The science of guessing: Analyzing an anonymized corpus of 70 million passwords,” in *Proc. IEEE Symp. Secur. Privacy*, 2012, pp. 538–552.
[34] A. Juels and R. L. Rivest, “Honeywords: Making password-cracking detectable,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 145–160.
[35] M. Bellare, D. Pointcheval, and P. Rogaway, “Authenticated key exchange secure against dictionary attacks,” in *Proc. 19th Int. Conf. Theory Appl. Cryptographic Techn.*, 2000, pp. 139–155.
[36] S. Halevi and H. Krawczyk, “Public-key cryptography and password protocols,” *ACM Trans. Inf. Syst. Secur.*, vol. 2, pp. 230–268, 1999.
[37] J. Liu, Y. Yu, and F.-X. Standaert, “Small tweaks do not help: Differential power analysis of milenage implementations in 3G/4G USIM cards,” in *Proc. 20th Eur. Symp. Res. Comput. Secur.*, 2015, pp. 468–480.
[38] X. Leroy, “Smart card security from a programming language and static analysis perspective,” 2013. [Online]. Available: http://pauillac.inria.fr/ ~ xleroy/talks/language-security-etaps03.pdf
[39] K. Nohl, D. Evans, S. Starbug, and H. Plötz, “Reverse-engineering a cryptographic RFID tag,” in *Proc. 17th Conf. Secur. Symp.*, 2008, pp. 185–193.
[40] B. Chen and W. Kuo, “Robust smart-card-based remote user password authentication scheme,” *Int. J. Commun. Syst.*, vol. 27, no. 2, pp. 377–389, 2014.
[41] S. Kumari and M. K. Khan, “Cryptanalysis and improvement of a robust smart-card-based remote user password authentication scheme,” *Int. J. Commun. Syst.*, vol. 27, no. 12, pp. 3939–3955, 2014.
[42] F. Hao, “On robust key agreement based on public key authentication,” in *Proc. 14th Int. Conf. Financial Cryptography Data Secur.*, 2010, pp. 383–390.
[43] D. Sun, J. Huai, J. Sun, J. Li, and Z. Feng, “Improvements of Juang et al.'s password-authenticated key agreement scheme using smart cards,” *IEEE Trans. Ind. Electron.*, vol. 56, no. 6, pp. 2284–2291, Jun. 2009.
[44] D. Wang, C. G. Ma, and P. Wu, “Secure password-based remote user authentication scheme with non-tamper resistant smart cards,” in *Proc. 26th Annu. IFIP WG 11.3 Conf. Data Appl. Secur. Privacy*, 2012, pp. 114–121.
[45] C. T. Li and C. Lee, “A robust remote user authentication scheme using smart card,” *Inf. Tech. Control*, vol. 40, no. 3, pp. 236–245, 2011.
[46] D. Wang and P. Wang, “On the anonymity of two-factor authentication schemes for wireless sensor networks: Attacks, principle and solutions,” *Comput. Netw.*, vol. 73, pp. 41–57, 2014.
[47] D. Wang, Q. Gu, H. Cheng, and P. Wang, “The request for better measurement: A comparative evaluation of two-factor authentication schemes,” in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, 2016, pp. 475–486.
[48] H. Krawczyk, “HMQV: A high-performance secure Diffie-Hellman protocol,” in *Proc. 25th Annu. Int. Conf. Advances Cryptology*, 2005, pp. 546–566.
[49] I. Liao, C. Lee, and M. Hwang, “A password authentication scheme over insecure networks,” *J. Comput. Syst. Sci.*, vol. 72, no. 4, pp. 727–740, 2006.
[50] E. Bresson, O. Chevassut, and D. Pointcheval, “Security proofs for an efficient password-based key exchange,” in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, 2003, pp. 41–50.
[51] X. Li, J. Niu, M. K. Khan, and J. Liao, “An enhanced smart card based remote user password authentication scheme,” *J. Netw. Comput. Appl.*, vol. 36, no. 5, pp. 1365–1371, 2013.
[52] K. Xue, P. Hong, and C. Ma, “A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture,” *J. Comput. Syst. Sci.*, vol. 80, no. 1, pp. 195–206, 2014.
[53] R. Martin, “Amid widespread data breaches in China,” Dec. 2011. [Online]. Available: http://www.techinasia.com/alipay-hack/
[54] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, “Targeted online password guessing: An underestimated threat,” in *Proc. ACM Conf. Comput. Commun. Secur.*, 2016, pp. 1–13, doi: 10.1145/2976749.2978339.
[55] H. Wimberly and L. Liebrock, “Using fingerprint authentication to reduce security: An empirical study,” in *Proc. IEEE Symp. Secur. Privacy*, 2011, pp. 32–46.
[56] V. Odelu, A. Das, and A. Goswami, “A secure biometrics-based multi-server authentication protocol using smart cards,” *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 9, pp. 1953–1966, Sep. 2015.
[57] Q. Jiang, J. Ma, G. Li, and X. Li, “Improvement of robust smart-card-based password authentication scheme,” *Int. J. Commun. Syst.*, vol. 28, no. 2, pp. 383–393, 2015.
[58] T.-T. Truong, M.-T. Tran, A.-D. Duong, and I. Echizen, “Chaotic Chebyshev polynomials based remote user authentication scheme in client-server environment,” in *Proc. 30th IFIP Int. Inf. Secur. Conf.*, 2015, pp. 479–494.
[59] S. Islam, “Design and analysis of an improved smartcard-based remote user password authentication scheme,” *Int. J. Commun. Syst.*, vol. 29, no. 11, pp. 1708–1719, 2016.
[60] M. Scott, N. Costigan, and W. Abdulwahab, “Implementing cryptographic pairings on smartcards,” in *Proc. 8th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2006, pp. 134–147.
[61] M. Scott, “Miracl library,” Shamus Softw. Ltd. [Online]. Available: http://www.shamus.ie/index.php?page=home

Ding Wang received the BS degree in information security from Nankai University, in 2008. Currently, he is working toward the PhD degree at Peking University. He received twice the Top-10 Distinguished Academic Fellow Award from the University and authored two “ESI highly cited papers”. He has been invited as the reviewer and TPC member for over 40 international journals and security conferences. His research interests focus on password cryptography. He is a student member of the IEEE.

Ping Wang received the BS degree from the University of Electronic Science and Technology of China, in 1983, and the PhD degree from the University of Massachusetts, in 1996. Now he is a professor with Peking University. He served as TPC co-chairs of several security conferences. His research interests include system security and distributed computing. He was co-recipient of three Best Paper Awards from different professional societies. He is a senior member of the IEEE.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2605087, IEEE Transactions on Dependable and Secure Computing

APPENDIX A
A COMPARATIVE EVALUATION OF EXISTING TWO-FACTOR AUTHENTICATION SCHEMES

To demonstrate the effectiveness of our framework in practice, we provide a comparative evaluation of 67 two-factor typical schemes by assessing whether the twelve criteria proposed in Section II-B have been met under the security model proposed in Section II-A. Particularly, among these 67 schemes, four were designed before 2004 where the tamper-resistance assumption of the smart cards are generally made. As expected, these early four schemes perform poorly under our new security model (see the bottom of Table A.1), while the recent schemes (e.g., [1]–[6]) generally perform much better. There is a trend that, under our evaluation framework, more recent the scheme is, more desirable the scheme will be.

However, this trend would not be obvious had these 67 schemes been assessed by the existing evaluation frameworks (i.e., [7]–[10]). More specifically, the criteria set in [7], essentially, only includes our criteria C2-C5, and thus one will not see the differences between the schemes proposed in 2010 and the schemes proposed in 2015; The criteria set in [8], [9] concerns "protocol efficiency" and most importantly, no security model is explicitly defined in [8], [9], all this would make these two frameworks virtually impossible to be decidable when assessing a scheme; Using the criteria set in [10] will not reveal the critical "usability-security tension" (discussed later), because it misses the desirable property "freely password change". All in all, the trend revealed by our framework partially demonstrates the soundness of our evaluation framework.

From the microcosmic point of view, one can see that each criterion is *satisfied* by at least 15 schemes and at the same time, it is *unmet* by at least 7 schemes. *This implies the necessity of each of the twelve criteria.* In addition, there is no scheme that can fulfill all the twelve criteria— the only scheme that can achieve eleven criteria is proposed by Odelu et al. in 2015 [4]. *This suggests the comprehensiveness of our criteria set. This also highlights the needs for more research efforts to design a better scheme.* With a careful examination, one can observe that this scheme suffers from the same "usability-security tension" with other latest schemes (e.g., [2], [3], [5]): the criterion C2 (or C9) and C4 cannot be achieved at the same time. *This suggests the necessity of the separation of C4 from C5*, in contrast to the framework in [7].

It is also worth noting that, in selecting a particular two-factor scheme for inclusion in the comparison Table A.1, we do not necessarily endorse it as better than alternatives that are not included in the table—merely because of that it is reasonably representative, or illuminates in some way what category (from a point view of the development tree where a specific scheme lies, see the history tree Fig. 2 in Section II) it belongs to can achieve. In addition, here we mainly focus on schemes for the single-server architecture because: (1) different architectures/environments may involve quite different attacking vectors and security models, and thus fair comparison is virtually impossible under a single security model; and (2) single-server-architecture-based schemes constitute the basis for schemes that are designed for other more complex architectures/environments (e.g., schemes for wireless sensor networks [11] and mobile networks [12]).

TABLE A.1. A COMPARATIVE EVALUATION OF TWO-FACTOR AUTHENTICATION SCHEMES

Scheme	Year	Ref.	No verifier table (C1)	Password friendly (C2)	No password exposure (C3)	No smart card loss problem (C4)	Resistance to known attacks (C5)	Sound repairability (C6)	Provision of key agreement (C7)	No clock synchronization (C8)	Timely typo detection (C9)	Mutual authentication (C10)	User anonymity (C11)	Forward secrecy (C12)
Lu *et al.*	2016	[13]	✓	✓	✓	·	·	✓	✓	✓	✓	✓	·	✓
Xie *et al.*	2016	[14]	✓	✓	✓	·	✓	·	✓	·	·	✓	·	✓
Islam *et al.*	2016	[15]	✓	✓	✓	·	✓	✓	✓	✓	✓	✓	✓	✓
Muhaya	2015	[1]	✓	✓	✓	·	✓	·	✓	✓	✓	✓	✓	·
Xu-Wu	2015	[2]	✓	·	✓	✓	·	✓	✓	✓	·	✓	✓	✓
Mishra *et al.*	2015	[3]	✓	✓	✓	·	✓	✓	✓	·	✓	✓	✓	·
Odelu *et al.*	2015	[4]	✓	✓	✓	·	✓	✓	✓	✓	✓	✓	✓	✓
Byun	2015	[5]	·	·	✓	✓	✓	✓	✓	✓	·	✓	✓	✓
Wang *et al.*	2015	[6]	✓	·	✓	✓	✓	✓	✓	·	✓	✓	✓	·
Truong *et al.*	2015	[16]	✓	✓	·	·	✓	·	✓	✓	✓	✓	·	✓
Djellali *et al.*	2015	[17]	✓	✓	·	·	✓	·	✓	✓	·	✓	✓	·
Mishra *et al.*	2015	[18]	✓	·	✓	·	✓	✓	✓	·	✓	✓	·	✓
Wu *et al.*	2015	[19]	·	✓	✓	·	·	·	✓	✓	·	✓	✓	✓
Chaudry *et al.*	2015	[20]	✓	✓	✓	·	·	✓	✓	·	✓	✓	·	·
Kumari *et al.*	2014	[21]	✓	✓	✓	·	✓	·	✓	✓	✓	✓	·	·
Chen *et al.*	2014	[22]	✓	·	·	·	✓	·	✓	·	·	✓	·	·
Tsai *et al.*	2013	[23]	✓	✓	✓	·	✓	·	✓	✓	·	✓	✓	✓
Li *et al.*	2013	[24]	✓	✓	·	·	✓	·	✓	·	✓	✓	·	✓
Kumari-Khan	2013	[25]	✓	·	·	·	✓	·	✓	·	✓	✓	✓	·
Sun-Cao	2013	[26]	✓	·	✓	·	✓	✓	✓	·	·	✓	✓	·
Lee-Liu	2013	[27]	✓	✓	✓	·	✓	·	✓	✓	·	✓	✓	·
Islam-Biswas	2013	[28]	·	·	·	✓	·	✓	✓	✓	·	✓	·	✓
Li-Zhang	2013	[29]	✓	·	·	✓	·	✓	✓	·	·	✓	✓	✓
Chang et al.	2013	[30]	✓	·	·	·	·	·	·	·	·	✓	·	·
Li	2013	[31]	✓	·	✓	·	·	✓	✓	✓	·	✓	✓	✓
Kim-Kim	2012	[32]	✓	·	·	✓	·	✓	✓	✓	·	✓	✓	·
Ramasamy-Muniyandi	2012	[33]	✓	✓	·	✓	·	·	·	·	·	·	·	·
Wu *et al.*	2012	[8]	✓	✓	✓	·	✓	✓	✓	✓	·	✓	✓	✓
Zhu	2012	[34]	✓	✓	✓	·	✓	✓	·	✓	·	✓	·	·
Wang-Ma	2012	[35]	✓	✓	✓	·	✓	✓	✓	·	✓	✓	✓	✓
Hsieh-Leu	2012	[36]	✓	·	·	·	✓	✓	·	·	·	✓	·	·
Wen-Li	2012	[37]	✓	·	·	·	·	✓	✓	·	·	✓	·	·
Wang *et al.*	2012	[38]	✓	✓	✓	·	✓	·	✓	✓	✓	✓	✓	✓
He *et al.*	2012	[39]	✓	·	✓	✓	✓	·	✓	·	·	✓	·	·
Wang *et al.*	2011	[40]	✓	✓	·	·	·	✓	✓	✓	·	✓	✓	✓
Chen *et al.*	2011	[41]	✓	✓	✓	·	·	·	✓	·	·	✓	·	·
Khan *et al.*	2011	[42]	✓	✓	✓	·	✓	✓	✓	·	✓	✓	·	·
Kim *et al.*	2011	[43]	✓	·	·	✓	✓	✓	✓	·	·	✓	·	·
Awasthi *et al.*	2011	[44]	✓	·	·	·	·	·	·	·	·	✓	·	·
Li-Lee	2011	[45]	✓	·	✓	·	·	✓	✓	✓	·	✓	·	·
Sood *et al.*	2011	[46]	✓	✓	·	·	·	✓	✓	·	✓	·	·	·
Li *et al.*	2010	[47]	✓	✓	✓	✓	·	✓	✓	✓	·	✓	✓	·
Tsai *et al.*	2010	[48]	✓	✓	·	·	✓	·	✓	·	·	✓	✓	·
Song	2010	[49]	✓	·	·	·	✓	·	✓	·	·	✓	·	·
Sood *et al.*	2010	[50]	✓	✓	·	✓	✓	·	✓	·	·	·	·	·
Yeh *et al.*	2010	[51]	✓	✓	·	·	✓	✓	✓	·	·	✓	·	·
Horng *et al.*	2010	[52]	✓	·	✓	✓	·	·	✓	✓	·	✓	✓	✓
Sun *et al.*	2009	[53]	✓	✓	✓	·	✓	✓	✓	✓	·	✓	·	✓
Hsiang-Shi	2009	[54]	✓	✓	·	·	✓	✓	·	·	·	✓	·	·
Xu *et al.*	2009	[55]	✓	·	·	·	·	·	✓	·	·	✓	·	✓
Kim-Chung	2009	[56]	✓	✓	·	·	·	·	·	·	✓	✓	·	·
Chung *et al.*	2009	[57]	✓	·	✓	✓	✓	✓	✓	✓	·	✓	·	✓
Ramasamy	2009	[58]	✓	·	·	✓	✓	✓	·	·	·	✓	·	·
Yang *et al.*	2008	[7]	✓	✓	✓	·	✓	·	✓	✓	·	✓	·	✓
Juang *et al.*	2008	[59]	✓	·	✓	✓	·	✓	✓	✓	·	✓	·	·
Chen *et al.*	2008	[60]	✓	·	✓	·	·	·	·	✓	·	✓	·	·
Wang *et al.*	2007	[61]	·	✓	✓	·	·	·	✓	·	✓	✓	·	·
Wang *et al.*	2007	[62]	✓	·	✓	·	·	✓	✓	✓	·	✓	·	·
Liao *et al.*	2006	[9]	✓	✓	✓	·	·	·	✓	·	·	✓	·	✓
Lee et al.	2005	[63]	✓	·	·	·	·	·	·	·	·	✓	·	·
Fan et al.	2005	[64]	✓	·	·	✓	✓	✓	·	✓	·	✓	·	·
Lu-Cao	2005	[65]	✓	·	·	✓	✓	·	·	·	·	·	·	·
Yoon *et al.*	2004	[66]	✓	✓	✓	·	·	✓	·	·	·	✓	·	·
Ku *et al.*	2004	[67]	✓	✓	✓	·	·	✓	·	·	·	✓	·	·
Wu-Chieu	2003	[68]	✓	✓	·	·	·	·	·	·	·	·	·	·
Awasthi-Lal	2003	[69]	·	·	·	·	·	·	·	·	·	·	·	·
Sun	2000	[70]	·	·	·	·	·	·	·	·	·	·	·	·
Hwang-Li	2000	[71]	·	·	·	·	·	·	·	·	·	·	·	·

*The present authors have contributed to the following schemes: Wang et al. [6], Wang et al. [38] and Wang-Ma [35]. Interested readers may verify that we have evaluated them impartially.

Finally, during the past five years the present authors have examined more than three hundred two-factor schemes, including over 170 ones for the single server architecture (see

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2605087, IEEE Transactions on Dependable and Secure Computing

some of our results [72]–[74]), over 100 ones for the multi-server architecture (see some of our results [75]), over 60 ones for mobile networks (see some of our results [76], [77]) and over 50 for the wireless sensor networks (see some of our results [78], [79]). Based on our past cryptanalysis experience, we believe that (1) whenever a specific scheme is identified by us to be *unable* to achieve some criteria, this is sufficiently definite to be true; and (2) when a specific scheme is identified by us to be able to achieve some criteria, this is highly likely to be true, while there may be some probability that we have missed some attacking modes. For instance, we have found that there are at least 8 vastly different attacking modes (e.g., attacking by returning back the extracted card and attacking by exploiting the first/second/third protocol flow) when investigating whether a scheme can resist offline password guessing attack in case the adversary has got access to victim's card, while this only constitutes part of the total work involved in evaluating just a single criteria C4. *This shows the great difficulty and the mount of manual work entailed when constructing a table like A.1.*

APPENDIX B
WIDE APPLICABILITY OF "FUZZY-VERIFIER" + "HONEYWORDS"

In this Section, we use representative schemes [23], [80], [81] as case studies to demonstrate exactly how our "fuzzy-verifier" and "honeywords" can be integrated into other schemes (and even schemes for other architectures). More specifically, we employ two typical schemes, i.e., Tsai et al.'s scheme [23] and Xue et al.'s scheme [80], to our approach can be integrated into two-factor authentication schemes for the single-server architecture and multi-server environment, respectively. In addition, we also use Odelu et al.'s scheme [81] to briefly show the applicability of our approach to three-factor authentication schemes. After our integration, one can confirm that existing usability-security conflicts in these schemes would be well eliminated. This shed light on how to eliminate the usability-security dilemma in various other schemes (e.g., [7], [8], [53], [82]).

A. Integration into Tsai et al.'s scheme

Tsai et al.'s scheme [23] has two versions: a two-factor one and a three-factor one. Here we mainly focus on the two-factor version. Their scheme consists of five phases, namely, parameter generation, registration, pre-computation, login and password update. Readers are referred to [23] for details of this scheme, but for ease of comprehension, we will follow its notations as closely as possible.

This scheme exhibits many desirable features over existing schemes, such as user anonymity, high efficiency and formal security proofs, yet it was recently found vulnerable to the smart card loss problem (i.e., unable to achieve C4) [72]. The key point is that, this scheme allows users to change their passwords freely and locally (i.e., able to achieve C2), yet there is no verification of the old password before the update of new password. If an attacker $\mathcal{A}$ manages to gain temporary access to the smart card of legitimate user U_i (note that this is a quite realistic assumption in practice), she can easily change the password of user U_i without any obstacle.

As revealed in [72] and further investigated in Sec. V-A, there are some subtleties and tricks in coping with this problem and before this work, there is no existing solution. Fortunately, our proposed "fuzzy-verifier" and "honeywords" can be integrated into this scheme as follows:

(1) In the registration phase, besides computing $V = h(ID_i\|x) \oplus h(PW_i\|b)$, the server further computes a fuzzy-verifier $A_i = h((h(\boldsymbol{ID}_i) \oplus h(PW_i\|b)) \bmod n_0)$ for U_i, and also stores both A_i and $A_i \oplus a_i$ in U_i's smart card, where $h(\cdot)$ is a one-way hash function and a_i is a random number. S stores $\{\boldsymbol{ID}_i, a_i,$ `Honey_List`$=$ NULL$\}$ in its backend database. The other parts remain the same as in [23].
(2) In the login phase, after the user U_i keys her identity ID_i and password PW_i, the card computes $A_i^* = h((h(\boldsymbol{ID}_i) \oplus h(PW_i\|b)) \bmod n_0)$ and verifies the validity of A_i^* by checking whether A_i^* equals to the stored A_i. If the verification holds, it implies the input $\boldsymbol{ID}_i$ and PW_i are valid with a probability of $\frac{n_0-1}{n_0}(\approx \frac{99.61}{100}$, when n_0=2^8). Otherwise, the smart card rejects. If A_i^* equals A_i, the smart card computes $a_i = (A_i \oplus a_i) \oplus A_i$ and $C_1 = (ID_i\| a_\mathbf{i}\|h(\boldsymbol{ID}_\mathbf{i}\|x)\ \|(h(ID_i\|x) \oplus N_{C2})) \oplus h_1(c)$. The other parts remain the same as in [23].
(3) In Step 3 of the login phase, after the server S receives the login request $\{C_1, e\}$ from U_i, S derives $ID_i'\|a_i'\|h(ID_i\|x)'$ from C_1, and checks whether a_i' equals the stored a_i. S rejects if they are not equal. Otherwise, S checks whether the derived $h(ID_i\|x)'$ equals the computed $h(ID_i\|x)$. If they are equal, S proceeds to the next step. If they are unequal, S now knows that $a_i' = a_i$ but $h(ID_i\|x)' \neq h(ID_i\|x)$, implying that there is a $1/2^{n_0}$ probability that U_i's card has been corrupted. Accordingly, S performs either (1) inserts $h(ID_i\|x)'$ into `Honey_List` when there are less than m_0 (e.g., $m_0 = 10$) items in `Honey_List`; or (2) suspends U_i's card (i.e., when there are m_0 items in `Honey_List`) until U_i re-registers. The other parts remain the same as in [23].
(4) In the password-change phase, the user U_i keys *twice* her identity ID_i and password PW_i. If U_i accidentally keys two unmatched (ID_i, PW_i) pairs, which is quite realistic in mobile device use cases, the card rejects. If they are match, the card computes $A_i^* = h((h(\boldsymbol{ID}_i) \oplus h(PW_i\|b)) \bmod n_0)$ and verifies the validity of A_i^* by checking whether A_i^* equals to the stored A_i. If the verification holds, it implies the input $\boldsymbol{ID}_i$ and PW_i are valid with a probability of $\frac{n_0-1}{n_0}(\approx \frac{99.61}{100}$, when n_0=2^8). Otherwise, the smart card rejects. Then, the card asks the user to submit a new password PW_i^{new} and computes $V^{new} = V \oplus h(PW_i\|b) \oplus h(PW_i^{new}\|b)$, $A_i^{new} = h((h(\boldsymbol{ID}_i) \oplus h(PW_i^{new}\|b)) \bmod n_0)$ and $a_i^{new} = (A_i \oplus a_i) \oplus A_i \oplus A_i^{new}$. Then, smart card updates the values of V, A_i and a_i with V^{new}, A_i^{new} and a_i^{new}, respectively.

Our above amendments are essentially based on the idea illustrated in Sec. V of the main text: we first force the attacker $\mathcal{A}$ to *has to* launch an online password guessing attack by interacting with S in order to determine the exactly correct password, and then design ways to enable S to *timely* detect the event that the parameters in U_i's card have been extracted and used by $\mathcal{A}$ to perform an online guessing attack. While there may be other ways to conquer this usability-security tension,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2605087, IEEE Transactions on Dependable and Secure Computing

we for the first time show the potential and provide a concrete solution to conquer this usability-security tension in [23].

B. Integration into Xue et al.'s scheme

Xue et al.'s scheme [80] is designed for the multi-server architecture, which is suitable for environments where users need to access more than one service server but only maintain one password and one smart card. Their scheme involves three participants: user U_i, service server S_j and control server CS. It consists of three phases (i.e., registration, login and authentication) and two phases (i.e., password update and dynamic identity update). Readers are referred to [80] for details of this scheme, but for ease of comprehension, we will follow its notations as closely as possible.

In U_i's card memory, there are two parameters stored: a random number b and $C_i = h(ID_i \| A_i) = h(ID_i \| h(b \| PW_i))$. These two parameters together can be used to support the property "timely typo detection" (i.e., C9). One can see that, if an adversary $\mathcal{A}$ obtains the card and extracts these two parameters, $\mathcal{A}$ can determine U_i's password PW_i as follows:

Step 1. Guesses the value of ID_i to be ID_i^* from dictionary space $\mathcal{D}_{id}$ and the value of PW_i to be PW_i^* from dictionary space $\mathcal{D}_{pw}$;
Step 2. Computes $C_i^* = h(ID_i^* \| A_i) = h(ID_i^* \| h(b \| PW_i^*))$, where b is revealed from U_i's card;
Step 3. Checks the correctness of (ID_i^*, PW_i^*) by comparing if the computed C_i^* equals the extracted C_i;
Step 4. Repeats Step 1, 2 and 3 of this procedure until the correct value of (ID_i^*, PW_i^*) is found.

The time complexity of the above attacking procedure is $\mathcal{O}(|\mathcal{D}_{id}| * |\mathcal{D}_{pw}| * 2T_H$, where T_H is the running time for Hash operation. In reality, the dictionary size is very restricted, e.g., $|\mathcal{D}_{id}| \leq |\mathcal{D}_{pw}| \leq 10^6$ [83], [84]. Further, according to the timings in Table B.1, $\mathcal{A}$ may determine the password in about 17.63 days on a common PC.

TABLE B.1. COMPUTATION EVALUATION OF RELATED OPERATIONS ON COMMON PCS

Experimental platform (common PCs)	Modular Exp. T_E ($\|n\| = 512$)	Hash operation T_H (SHA-1)	Other lightweight oper.(e.g., XOR, $\|$)
Intel T5870 2.00 GHz	2.573 ms	2.437 µs	0.011 µs
Intel i5750 2.66 GHz	2.106 ms	1.980 µs	0.009 µs
Pentium IV 3.06 GHz	1.676 ms	1.523 µs	0.008 µs

In addition, *user ID generally cannot be considered as a secret and actually, it is often publicly available*. Thus, there is a high probability for $\mathcal{A}$ to learn the user's identity ID_i other than guessing it. In this light, the above attack will be more practical. What's more, high performance computers are quite common those days and cheap cloud computing services are also easily available (e.g., Amazon EC2 [85]). All this indicates that the above attack is effective even if $\mathcal{A}$ has to figure out both ID_i and PW_i simultaneously.

To conquer this vulnerability while still preserving C9, the *definite* password verifier $C_i = h(ID_i \| h(b \| PW_i))$ shall be changed to a *fuzzy* verifier $FC_i = h(ID_i \| h(b \| PW_i) \texttt{mod}\ n_0)$. Furthermore, some "honeywords" shall be kept by CS to detect the user card breach event. More specifically, our proposed "fuzzy-verifier" and "honeywords" can be integrated into this scheme as follows:

(1) In the registration phase, U_i does not compute C_i but instead computes a fuzzy-verifier $FC_i = h(h(ID_i \| h(b \| PW_i)) \texttt{ mod } n_0)$, and stores FC_i and $FC_i \oplus a_i$ in U_i's smart card, where $h(\cdot)$ is a one-way hash function and a_i is a random number. The control server CS stores $\{\boldsymbol{ID}_i, a_i, \texttt{Honey\_List}=\text{NULL}\}$ in its backend database. The other parts remain the same as in [23].
(2) In the login phase, after the user U_i keys her identity ID_i and password PW_i, the card computes $FC_i = h(h(ID_i^* \| h(b \| PW_i^*)) \texttt{ mod } n_0))$ and verifies the validity of FC_i^* by checking whether FC_i^* equals to the stored FC_i. If the verification holds, it implies the input $\boldsymbol{ID}_i$ and PW_i are valid with a probability of $\frac{n_0-1}{n_0} (\approx \frac{99.61}{100}$, when n_0=2^8). Otherwise, the smart card rejects. If FC_i^* equals FC_i, the smart card computes $a_i = (FC_i \oplus a_i) \oplus FC_i)$ and $CID_i = (ID_i \| a_i \| B_i) \oplus h(B_i \| N_{i1} \| TS_i \| \text{“}00\text{''})$. The other parts remain the same as in [80].
(3) In Step 3 of the login phase, after the control server CS receives the login request $\{F_i, P_{ij}, CID_i, G_i, PID_i, TS_i, J_i, K_i, L_i, M_i, PSID_j\}$ from S_j, CS derives $ID_i' \| a_i' \| B_i'$ from CID_i, and checks whether a_i' equals the stored FC_i. S rejects if they are not equal. Otherwise, CS checks whether the derived B_i' equals the computed $h(PID_i \| x)$. If they are equal, CS proceeds to the next step. If they are unequal, CS now knows that $a_i' = a_i$ but $h(PID_i \| x)' \neq h(PID_i \| x)$, implying that there is a $1/2^{n_0}$ probability that U_i's card has been corrupted. Accordingly, CS performs either (1) inserts $h(PID_i \| x)'$ into `Honey_List` when there are less than m_0 (e.g., $m_0 = 10$) items in `Honey_List`; or (2) suspends U_i's card (i.e., when there are m_0 items in `Honey_List`) until U_i re-registers. The other parts remain the same as in [80].
(4) In the password-change phase, the user U_i keys *twice* her identity ID_i and password PW_i. If U_i accidentally keys two unmatched (ID_i, PW_i) pairs, which is quite realistic in mobile device use cases, the card rejects. If they are match, the card computes $FC_i^* = h(h(ID_i^* \| h(b \| PW_i^*)) \texttt{ mod } n_0)$ and verifies the validity of FC_i^* by checking whether FC_i^* equals to the stored FC_i. If the verification holds, it implies the input $\boldsymbol{ID}_i$ and PW_i are valid with a probability of $\frac{n_0-1}{n_0} (\approx \frac{99.61}{100}$, when n_0=2^8). Otherwise, the smart card rejects. Then, the card asks the user to submit a new password PW_i^{new} and computes $FC_i^{new} = h(h(ID_i \| h(b \| PW_i^{new})) \texttt{ mod } n_0)$, $D_i^{new} = D_i \oplus h(h(\boldsymbol{ID}_i \| b) \oplus h(b \| PW_i) \oplus h(b \| PW_i^{new}))$ and $a_i^{new} = (A_i \oplus a_i) \oplus A_i \oplus A_i^{new}$. Then, smart card updates the values of FC_i, D_i and a_i with FC_i^{new}, D_i^{new} and a_i^{new}, respectively.

C. Integration into Odelu et al.'s scheme

Odelu et al.'s scheme [81] is three-factor scheme designed for the multi-server architecture. As we have said earlier, this kind of scheme is suitable for environments where users need to access more than one service server but only maintain one password and one smart card (as well as her fingerprint). Their scheme involves three participants: user U_i, service server S_j and registration center RC (which serves as the same role of the control server CS in Xue et al.'s scheme [80]). It consists of four phases (i.e., initialization, registration, login and authentication) and two phases (i.e., password update, and Revocation and re-registration). Readers are referred to [81]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2605087, IEEE Transactions on Dependable and Secure Computing

for details of this scheme, but for ease of comprehension, we will follow its notations as closely as possible.

In U_i's card memory, there are three parameters stored: an auxiliary string θ_i, $s_i = H(k_i \| ID_i \| H(pw_i \| \sigma_i))$ and $z_i = k_i \oplus H(pw_i \| \sigma_i)$. It is not difficult to see that, Odelu et al.'s scheme [81] cannot provide truly "three-factor security", which is the most essential goal of a three-factor scheme. More specifically, U_i's password factor can be offline guessed by using s_i as a comparison target if the other two authentication factors (i.e., smart card and biometric) have been breached. Odelu et al. have realized this issue and pointed out that their scheme "can achieve the three-factor authentication by removing the hash value s_i from the smart-card" and "In that case, the password change will not be possible locally."

Fortunately, our "fuzzy-verifier" and "honeywords" can be integrated into this scheme to ensure that three-factor security can still be achieved while preserving the property of "local and secure password change". The details are as follows:

(1) In the registration phase, the registration center RC computes s_i as $s_i = H((k_i \| ID_i \| H(pw_i \| \sigma_i)) \bmod n_0)$, and stores s_i and $s_i \oplus a_i$ in U_i's smart card, where $H(\cdot)$ is a one-way hash functions. RC stores $\{ID_i, a_i, H(ID_i \| k), r_i,$ Honey_List=NULL$\}$ in its backend database. The other parts remain the same as in [81].

(2) In the login phase, after the user U_i keys her identity ID_i and password PW_i and imprints her personal biometrics B_i' at the sensor. Then, smart card computes $\sigma_i' = Rep(B_i', \theta_i)$ and $k_i' = z_i' \oplus H(pw_i' \| \sigma_i')$ and checks whether $s_i' = H((k_i' \| ID_i \| H(pw_i' \| \sigma_i')) \bmod n_0)$ equals the stored s_i. If the verification holds, it implies the input PW_i is valid with a probability of $\frac{n_0-1}{n_0} (\approx \frac{99.61}{100}$, when n_0=2^8). Otherwise, the smart card rejects. If s_i' equals s_i, the smart card computes $C_1 = E_{K_{1x}}[ID_i \| a_i \| k_i \| SID_j \| s_j \| n_1]$. The other parts remain the same as in [81].

(3) In Step AK2 of the Authentication phase, after the the registration center RC receives the login request $\{C_1, X, h_1, C_2, h_2\}$ from S_j, RC derives $ID_i' \| a_i' \| k_i'$ from C_1, and checks whether a_i' equals the stored a_i. RC rejects if they are not equal. Otherwise, RC checks whether the derived k_i' equals the computed $k_i = H(ID_i \| k \| r_i \| H(ID_i \| k))$. If they are equal, RC proceeds to the next step. If they are unequal, RC now knows that $a_i' = a_i$ but $k_i' \neq H(ID_i \| k \| r_i \| H(ID_i \| k))$, implying that there is a $1/2^{n_0}$ probability that U_i's card has been corrupted. Accordingly, RC performs either (1) inserts k_i' into Honey_List when there are less than m_0 (e.g., $m_0 = 10$) items in Honey_List; or (2) suspends U_i's card (i.e., when there are m_0 items in Honey_List) until U_i re-registers. The other parts remain the same as in [81].

(4) In the password-change phase, the user U_i keys *twice* her identity ID_i and password PW_i. If U_i accidentally keys two unmatched (ID_i, PW_i) pairs, which is quite realistic in mobile device use cases, the card rejects. If they are match, U_i inputs her biometric B_i' and the card computes $\sigma_i' = Rep(B_i', \theta_i)$, $s_i^* = H((k_i \| ID_i \| H(pw_i \| \sigma_i')) \bmod n_0)$ and verifies the validity of s_i^* by checking whether s_i^* equals to the stored s_i. If the verification holds, it implies the input PW_i is valid with a probability of $\frac{n_0-1}{n_0} (\approx \frac{99.61}{100}$, when n_0=2^8). Otherwise, the smart card rejects. Then, the card asks the user to submit a new password PW_i^{new} and computes $s_i^{new} = H((k_i \| ID_i \| H(pw_i^{new} \| \sigma_i)) \bmod n_0)$, $z_i^{new} = z_i \oplus H(PW_i \| \sigma_i) \oplus H(PW_i^{new} \| \sigma_i)$ and $a_i^{new} = (s_i \oplus a_i) \oplus s_i \oplus s_i^{new}$. Then, smart card updates the values of s_i, z_i and a_i with s_i^{new}, z_i^{new} and a_i^{new}, respectively.

APPENDIX C
FORMAL SECURITY ANALYSIS OF OUR SCHEME

A. Proof of Theorem 1

Proof. In the proof below, we do not consider forward-secrecy for simplicity. We incrementally define a sequence of games starting at the real attack game G_0 and ending up with G_8. For each game G_n (n=0,1, …, 8), we define the following events:

- $\mathbf{Succ_n}$ occurs if $\mathcal{A}$ correctly guesses the bit c involved in the Test-query.
- $\mathbf{AskPara_n}$ occurs if $\mathcal{A}$ correctly computes the parameter k by asking a hash query $\mathcal{H}_0$ on $b \| PW_i$ or $x \| ID_i \| T_{reg}$.
- $\mathbf{AskAuth_n}$ occurs if $\mathcal{A}$ correctly computes the parameter k and asks a hash query $\mathcal{H}_1$ (or $\mathcal{H}_2$) on $ID_i \| ID_S \| Y_1 \| C_2 \| k \| K$, where K is K_U or K_S.
- $\mathbf{AskH_n}$ occurs if the adversary asks a hash query $\mathcal{H}_i$ $(i = 1, 2, 3)$ on $ID_i \| ID_S \| Y_1 \| C_2 \| k \| K$, where K is K_U or K_S.

Game $\mathbf{G_0}$: This game corresponds to the real attack, in the random oracle model. By definition, we have

$$\mathrm{Adv}_{\mathcal{P}}^{\mathrm{ake}}(\mathcal{A}) = 2\Pr[\mathrm{Succ}_0] - 1. \quad (1)$$

Game $\mathbf{G_1}$: In this game, we simulate the hash oracles $\mathcal{H}_i$(i=0,1,2 and 3, but also four additional hash functions $\mathcal{H}_i'$ that will appear in Game $\mathbf{G_7}$) as usual by maintaining a hash list $\Lambda_{\mathcal{H}}$(and another list $\Lambda_{\mathcal{A}}$ containing the hash-queries asked by the adversary itself). We also simulate all the instances, as the real players would do, for the Send-queries and for the Execute, Reveal, Corrupt and Test-queries (see Figure C.1).

From this simulation, one can easily see that this game is perfectly indistinguishable from the real attack. Hence,

$$\left|\Pr[\mathrm{Succ}_1] - \Pr[\mathrm{Succ}_0]\right| = 0 \quad (2)$$

Game $\mathbf{G_2}$: For an easier analysis, in this game, we simulate all oracles as in game $\mathbf{G_1}$ except that we cancel games in which some (unlikely) collisions appear:

- collisions on the partial transcripts $((C_1, M_i, CAK_i, CID_i), (C_2, C_3), C_4)$. Note that transcripts involve at least one honest party, and thus one of C_1 or C_2 is truly uniformly distributed;
- collisions on the output of hash queries.

Both probabilities are bounded by the birthday paradox:

$$\left|\Pr[\mathrm{Succ}_2] - \Pr[\mathrm{Succ}_1]\right| \leq \frac{(q_{send} + q_{exe})^2}{2p} + \frac{q_h^2}{2^{l+1}} \quad (3)$$

where $l = min\{l_i\}, i = 0, 1, 2, 3$.

Game $\mathbf{G_3}$: We define game $\mathbf{G_3}$ by aborting the game wherein the adversary may have lucky in guessing the correct authenticator C_3 or C_4 (that is, without asking the corresponding hash query $\mathcal{H}_1$ or $\mathcal{H}_2$). Since C_1 and C_2 did not appear in a previous session (since the Game $\mathbf{G_2}$), this happens only if

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2605087, IEEE Transactions on Dependable and Secure Computing

For a hash-query $\mathcal{H}_i(q)$ or $\mathcal{H}_i'(q)$ (with $i \in \{0,1,2,3\}$), such that a record (i,q,r) appears in $\Lambda_{\mathcal{H}}$, the answer is r. Otherwise the answer r is defined according to the following rule:

▶**Rule** $\mathcal{H}^{(1)}$ — Choose a random element $r \in \{0,1\}^{1_i}$.

The record (i,q,r) is added to $\Lambda_{\mathcal{H}}$. If the query is directly asked by the adversary, one adds (i,q,r) to Λ .

We answer to the Send-queries to the client as follows:

— A Send (U^i, Start) -query is processed according to the following rule:

▶**Rule U1**$^{(1)}$ — Choose $\theta \in_R \mathbb{Z}_p^*$ and compute $C_1 = g^\theta$, $Y_1 = y^\theta$, $k = N_i \oplus \mathcal{H}_0(b \| PW_i)$, $M_i = \mathcal{H}_0(Y_1 \| k \| CAK_i \| CID_i)$, $CAK_i = (a_i \| k) \oplus \mathcal{H}_0(Y_1 \| C_1)$, and $CID_i = ID_i \oplus \mathcal{H}_0(C_1 \| Y_1)$.

Then the query is answered with (C_1, M_i, CID_i, CAK_i), and the client instance goes to an expected state.

— If the client instance U^i is in an expected state, a query Send $(U^i, (C_2, C_3))$ is processed by computing the session key and producing an authenticator. We apply the following rules:

▶**Rule U2**$^{(1)}$ — Compute $K_U = C_2^\theta$ and $C_3^* = \mathcal{H}_1(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_U)$.
Reject if the computed C_3^* is not equal to the received C_3. Otherwise moves on.

▶**Rule U3**$^{(1)}$ — Compute the authenticator $C_4 = \mathcal{H}_2(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_U)$ and the session key $sk_U = \mathcal{H}_3(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_U)$.

Finally the query is answered with C_4, the client instance accepts and terminates. Our simulation also adds $((C_1, M_i, CID_i, CAK_i), (C_2, C_3), C_4)$ to Λ_Ψ. The variable Λ_Ψ keeps track of the exchanged messages.

We answer to the Send-queries to the server as follows:

— A Send $(S^j, (C_1, M_i, CID_i, CAK_i))$-query is processed according to the following rule:

▶**Rule S1**$^{(1)}$ — Compute $Y = C_1^x$ and $ID_i = CID_i \oplus \mathcal{H}_0(C_1 \| Y_1)$, and rejects ID_i is not valid.
Reject if the computed $M_i^* = \mathcal{H}_0(Y_1 \| k \| CID_i \| CAK_i)$ is not equal to the received M_i.
Compute $k = \mathcal{H}_0(x \| ID_i \| T_{reg})$ and $a_i' \| k' = CAK_i \oplus \mathcal{H}_0(Y_1 \| C_1)$.
Reject if $A_i' \neq$ stored A_i;
Suspend U_i if $|Honey\_list| \geq m_0 \wedge (k' \neq k) \wedge (a_i' = a_i)$;
Insert k' into $Honey\_list$ if $|Honey\_list| < m_0$.

▶**Rule S2**$^{(1)}$ — Choose a random exponent $\varphi \in ¢_q^*$;
Compute $K_S = C_1^\varphi$, $C_2 = g^\varphi$, and $C_3 = \mathcal{H}_1(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_S)$.

Finally, the query is answered with (C_2, C_3) and the server instance goes to an expected state.

— If the server instance S^j is in an expected state, a query Send (S^j, C_4) is processed according to the following rules:

▶**Rule S3**$^{(1)}$ — Compute $C_4^* = \mathcal{H}_2(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_S)$, and check whether $C_4^* = C_4$.

If the equality does not hold, the server instance terminates without acceptance.

If equality holds, the server instance accepts and goes on, applying the following rule:

▶**Rule S4**$^{(1)}$ — Compute the session key $sk_S = \mathcal{H}_3(ID_i \| ID_S \| Y_1 \| C_2 \| k \| K_S)$.

Finally, the server instance terminates.

An Execute (U^i, S^j) -query is processed using a successive simulations of the Send-queries: $(U,(C_1, M_i, CID_i)) \leftarrow$ Send (U^i, Start) , $(C_2, C_3) \leftarrow$ Send $(S^j, (C_1, M_i, CID_i, CAK_i))$ and $C_4 \leftarrow$ Send$(U^i, (C_2, C_3))$, and outputting the transcript $((C_1, M_i, CID_i, CAK_i), (C_2, C_3), C_4)$.

A Corrupt(U, a)-query returns the password PW_i if a=1; returns $\{N_i, A_i, b\}$ if a=2. (Since we do not consider forward secrecy in this proof, no Corrupt(S, 1) -query occurs.

A Reveal(I)-query returns the session key (sk_U or sk_S) computed by the instance I (if the latter has accepted).

A Test(I)-query first gets sk from Reveal(I), and flips a coin c. If c = 1, we return the value of the session key sk, otherwise we return a random value drawn from $\{0,1\}^{l_3}$.

Fig. C.1. Simulation of the queries in our scheme

the authenticator C_3 (or C_4) had been correctly guessed by $\mathcal{A}$ without asking $\mathcal{H}_1$ (or $\mathcal{H}_2$):

$$\left|\Pr[\text{Succ}_3] - \Pr[\text{Succ}_2]\right| \leq \frac{q_{send}}{2^l} \quad (4)$$

Game $\mathbf{G_4}$: We define game $\mathbf{G_4}$ by aborting the game wherein the adversary may have lucky in guessing the correct parameter k (i.e., without asking the corresponding query). We reach this aim by modifying the way the participants process the queries. We use the rule as follows:

- ▶ **Rule U3$^{(4)}$** – Look for a record $(0, *\|ID_i\|*, k)$ in $\Lambda_\mathcal{A}$. If such a record does not exist, we abort the game. Otherwise, compute $C_4 = \mathcal{H}_2(ID_i\|ID_S\|Y_1\| C_2\|k \|K_U)$ and the session key $sk_U = \mathcal{H}_3(ID_i\|ID_S\| Y_1\|C_2\|k\|K_U)$.
- ▶ **Rule S3$^{(4)}$** – computes $C_4^* = \mathcal{H}_2(ID_i\|ID_S\|Y_1\|C_2 \|k\|K_S)$ and then checks if C_4^* equals the received value of C_4. If this verification holds, the server looks for a record $(0, *, P_i)$ in $\Lambda_\mathcal{A}$ and a record $((C_1, M_i, CAK_i, CID_i), (C_2, C_3), C_4)$ in Λ_Ψ. If such records do not exist, we abort the game.

Since C_1 and C_2 did not appear in a previous session (since the Game $\mathbf{G_2}$), this happens only if the parameter k had been correctly guessed by the adversary without asking $\mathcal{H}_0$:

$$\left|\Pr[\text{Succ}_4] - \Pr[\text{Succ}_3]\right| \leq \frac{q_{send}}{2^{l_0}} \quad (5)$$

Game $\mathbf{G_5}$: We define this game by aborting the game wherein the adversary may have computed the correct parameter k and impersonate as a client or server. We reach this aim by modifying the way the participants process the queries. We use the rule as follows:

- ▶ **Rule U2$^{(5)}$** – Look for a record $(0, *\|ID_i\|*, k)$ in $\Lambda_\mathcal{A}$. If such a record exists, we abort the game. Otherwise, compute $K_U = (C_2)^u \bmod p$, $C_3^* = \mathcal{H}_1(ID_i\|ID_S\|Y_1 \|C_2\|k\|K_U)$, and compare C_3^* with the received C_3. If the equality does not hold, terminate without acceptance. Otherwise, move on.
- ▶ **Rule S3$^{(5)}$** – computes $C_4^* = \mathcal{H}_2(ID_i\|ID_S\|Y_1\| C_2\|k\|K_S)$ and then checks if C_4^* equals the received value of C_4. If this verification holds, the server looks for a record $(0, *, P_i)$ in $\Lambda_\mathcal{A}$ and a record $((C_1, M_i, CAK_i, CID_i), (C_2, C_3), C_4)$ in Λ_Ψ. If such records exist, we abort the game.

The two games $\mathbf{G_5}$ and $\mathbf{G_4}$ are perfectly indistinguishable unless the event $\mathbf{AskPara_5}$ occurs:

$$\left|\Pr[\text{Succ}_5] - \Pr[\text{Succ}_4]\right| \leq \Pr[\text{AskPara}_5] \quad (6)$$

To upper bound $\Pr[\text{AskPara}_5]$, the parameter k is assumed to be correctly computed by $\mathcal{A}$ in all the ensuing games.

Game $\mathbf{G_6}$: We define this game by aborting the game wherein the adversary may have computed the correct authenticator C_3 or C_4 (that is, by asking the corresponding hash query $\mathcal{H}_1$ or $\mathcal{H}_2$) and impersonate as a client or server. We reach this aim by modifying the way the participants process the queries. We use the rule as follows:

- ▶ **Rule U3$^{(6)}$** – Check if $(1, ID_i\|ID_S\|Y_1\|C_2\|k\|*, C_3) \in \Lambda_\mathcal{A}$. If it holds, we abort the game. Otherwise, the user goes on to compute C_4 and sk_U.
- ▶ **Rule S3$^{(6)}$** – computes $C_4^* = \mathcal{H}_2(ID_i\|ID_S\|Y_1\| C_2\|k\|K_S)$ and then checks if C_4^* equals the received value of C_4. If this verification holds, the server looks for a record $(0, *, P_i)$ or $(2, ID_i\|ID_S\|Y_1\|C_2\|k\|*, C_4)$ in $\Lambda_\mathcal{A}$, and a record $((C_1, CAK_i, CID_i), (C_2, C_3), C_4)$ in Λ_Ψ. If such records exist, we abort the game.

The two games $\mathbf{G_6}$ and $\mathbf{G_5}$ are perfectly indistinguishable unless event $\mathbf{AskAuth_6}$ occurs:

$$\left|\Pr[\text{Succ}_6] - \Pr[\text{Succ}_5]\right| \leq \Pr[\text{AskAuth}_6] \quad (7)$$

$$\left|\Pr[\text{AskPara}_6] - \Pr[\text{AskPara}_5]\right| \leq \Pr[\text{AskAuth}_6] \quad (8)$$

Game $\mathbf{G_7}$: In this game, we replace the random oracles $\mathcal{H}_i$ with the private oracles $\mathcal{H}_i'(i = 1, 2, 3)$:

$$C_3 = \mathcal{H}_1'(ID_i\|ID_S\|C_1\|C_2);$$
$$C_4 = \mathcal{H}_2'(ID_i\|ID_S\|C_1\|C_2);$$
$$sk_U = sk_S = \mathcal{H}_3'(ID_i\|ID_S\|C_1\|C_2)$$

As a result, the values of C_3, C_4, sk_U, sk_S are completely independent from k, K_U and K_S. $\mathbf{G_7}$ and $\mathbf{G_6}$ are indistinguishable unless the event $\mathbf{AskH_7}$ occurs:

$$\left|\Pr[\text{Succ}_7] - \Pr[\text{Succ}_6]\right| \leq \Pr[\text{AskH}_7] \quad (9)$$

$$\left|\Pr[\text{AskPara}_7] - \Pr[\text{AskPara}_6]\right| \leq \Pr[\text{AskH}_7] \quad (10)$$

$$\left|\Pr[\text{AskAuth}_7] - \Pr[\text{AskAuth}_6]\right| \leq \Pr[\text{AskH}_7] \quad (11)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2016.2605087, IEEE Transactions on Dependable and Secure Computing

Lemma 1: The probabilities of the events $\mathbf{Succ_7}$ and $\mathbf{AskPara_7}$ in this game can be up-bounded by:

$$\left|\Pr[\mathrm{Succ}_7]\right| = \frac{1}{2} \qquad \left|\Pr[\mathrm{AskPara}_7]\right| \le C' \cdot q_{send}^{s'} + \frac{q_{send}}{2^{l_0}} \quad (12)$$

Proof. In the game $\mathbf{G_7}$, the session keys are computed with private hash oracle unknown to $\mathcal{A}$, and thus $\Pr[\mathrm{Succ}_7] = \frac{1}{2}$.

Let us denote by $R(U)$ the set of (C_2, C_3) received by a client instance, and by $R(S)$ the set of C_4 used by a server instance. Since we have avoided the cases where $\mathcal{A}$ have been lucky in guessing k, $\mathcal{A}$ can correctly compute k with the help of either a Corrupt($I = U^i$, 1)-query or a Corrupt($I = U^i$, 2)-query, the probability of which is denoted by $\Pr[\mathrm{AskPara_7WithCorr_1}]$ and $\Pr[\mathrm{AskPara_7WithCorr_2}]$, respectively. As discussed in Sec. III of the main text, it is more desirable (realistic) to assume passwords to be Zipf-distributed [86] than to make the traditional, commonly used (yet unrealistic) assumption that passwords are uniformly distributed (see some notable literature [7], [87]–[90]). From an information theoretical point of view, since we have avoided collisions in the Game $\mathbf{G_2}$,

$$\begin{aligned}&\left|\Pr[\mathrm{AskPara_7WithCorr_1}]\right| \\ &= \Pr_{pw}[\exists C_3 \in R(U), (1, ID_i\|ID_S\|Y_1\|C_2\|k\|*, C_3) \in \Lambda_{\mathcal{A}}] \\ &+ \Pr_{pw}[\exists C_4 \in R(S), (0, *, P_i) \in \Lambda_{\mathcal{A}}] \le C' \cdot q_{send}^{s'} \qquad (13)\end{aligned}$$

$$\Pr[\mathrm{AskPara_7WithCorr_2}]\right| \le \tfrac{q_{send}}{2^{l_0}} \qquad (14)$$

Game $\mathbf{G_8}$: In this game, we simulate the executions using the random self-reducibility of the Diffie-Hellman problem [91], given one CDH instance (A, B). Note that, we do not need to know the values of θ and φ, since the values of K_U and K_S are no longer needed to compute the authenticators or session keys:

- ► **Rule U1$^{(8)}$** – chooses a random number $\alpha \in Z_p^*$ and computes C_1=A^α mod p, Y_1=y^α mod p, $k = \mathcal{H}_0(x\|ID_i\|T_{reg}) = N_i \oplus \mathcal{H}_0(b\|PW_i)$, $CID_i = ID_i \oplus \mathcal{H}_0(C_1\|Y_1)$, $CAK_i = (a_i\|k) \oplus \mathcal{H}_0(Y_1\|C_1)$ and $M_i = \mathcal{H}_0(Y_1\|k\|CID_i)$. Also add the record (C_1, α) in Λ_A.
- ► **Rule S2$^{(8)}$** – chooses a random number $\beta \in Z_p^*$, and computes $C_2 = B^\beta$ and $C_3 = \mathcal{H}'_1(ID_i\|ID_S\|C_1\|C_2)$. Also adds the record (C_2, β) in Λ_B.

$$\Pr[\mathrm{AskH}_7] = \Pr[\mathrm{AskH}_8] \qquad (15)$$

Remember that $\mathbf{AskH_8}$ means that the adversary $\mathcal{A}$ had queried the random oracles $\mathcal{H}_i (i = 1, 2, 3)$ on $(ID_i\|ID_S\| Y_1\|C_2\|*\|CDH(C_1, C_2))$. By picking randomly in the $\Lambda_{\mathcal{A}}$-list we can get the Diffie-Hellman secret value with probability $\frac{1}{q_h}$. This is a triple $(C_1, C_2, \mathrm{CDH}(C_1, C_2))$. We can then simply look in the lists Λ_A and Λ_B to find the values α and β such that $C_1 = A^\alpha$ and $C_2 = B^\beta$:

$$\mathrm{CDH}(C_1, C_2) = \mathrm{CDH}(A^\alpha, B^\beta) = \mathrm{CDH}(A, B)^{\alpha\beta}$$

and thus:

$$\left|\Pr[\mathrm{AskH}_8]\right| \le q_h \mathrm{Adv}_{\mathcal{P}}^{\mathrm{CDH}}(t') \qquad (16)$$

where $t' \le t + (q_{send} + q_{exe} + 1) \cdot \tau_e$.

Conclusion of the proof: By combining above equations, one gets the announced result. Firstly, from Eqs.(1)-(5) we get:

$$\left|\Pr[\mathrm{Succ}_4] - \Pr[\mathrm{Succ}_0]\right| \le \tfrac{(q_{send}+q_{exe})^2}{2p} + \tfrac{q_h^2}{2^{l+1}} + \tfrac{q_{send}}{2^l}.$$

Secondly, from Eqs.(6)-(9) we get:

$$\begin{aligned}\left|\Pr[\mathrm{Succ}_7] - \Pr[\mathrm{Succ}_4]\right| \le \Pr[\mathrm{AskPara}_5] + \Pr[\mathrm{AskAuth}_6] \\ +\Pr[\mathrm{AskH}_7].\end{aligned}$$

Thirdly, from the definition we know:

$$\Pr[\mathrm{AskAuth}_6] \le \Pr[\mathrm{AskH}_6].$$

Finally, based on Eqs.(10)-(16) we get:

$$\begin{aligned}\mathrm{Adv}_{\mathcal{P}}^{\mathrm{ake}}(\mathcal{A}) &= 2\Pr[\mathrm{Succ}_7] - 1 + 2(\Pr[\mathrm{Succ}_0] - \Pr[\mathrm{Succ}_7]) \\ &\le C' \cdot q_{send}^{s'} + 12 q_h \mathrm{Adv}_{\mathcal{P}}^{\mathrm{CDH}}(t') \\ &+ \frac{q_h^2 + 6q_{send}}{2^l} + \frac{(q_{send} + q_{exe})^2}{p},\end{aligned}$$

where we use the Zipf model of Tianya in [86], where C'= 0.062239 and s'=0.155478; $n_0 = 2^8$; $t' \le t + (q_{send} + q_{exe} + 1) \cdot \tau_e$ and $l = min\{l_i\}, i = 0, 1, 2, 3$. □

B. Proof of Theorem 2

Proof. The proof is similar to that of Theorem 1.

Firstly, we define an additional event:

– $\mathbf{Auth_n}$ occurs if $\mathcal{A}$ correctly guesses the authenticator C_3 or C_4 that will be accepted by the corresponding party and that has been built by the adversary herself in game $\mathbf{G_n}, n = 0, 1, \ldots, 7$.

Thus, we define

$$\mathrm{Adv}_{\mathcal{P}}^{\mathrm{auth}}(\mathcal{A}) = \Pr[\mathrm{Auth}_0]$$

Secondly, we use the same sequence of games presented in the previous section, and extend Eqs. (1)-(7) to obtain:

$$\begin{aligned}&\left|\Pr[\mathrm{Auth}_1] - \Pr[\mathrm{Auth}_0]\right| = 0 \\ &\left|\Pr[\mathrm{Auth}_2] - \Pr[\mathrm{Auth}_1]\right| \le \frac{(q_{send} + q_{exe})^2}{2p} + \frac{q_h^2}{2^{l+1}} \\ &\left|\Pr[\mathrm{Auth}_3] - \Pr[\mathrm{Auth}_2]\right| \le \frac{q_{send}}{2^l} \\ &\left|\Pr[\mathrm{Auth}_4] - \Pr[\mathrm{Auth}_3]\right| \le \frac{q_{send}}{2^l} \\ &\left|\Pr[\mathrm{Auth}_5] - \Pr[\mathrm{Auth}_4]\right| \le \Pr[\mathrm{AskPara}_5] \\ &\left|\Pr[\mathrm{Auth}_6] - \Pr[\mathrm{Auth}_5]\right| \le \Pr[\mathrm{AskAuth}_6] \le 2\Pr[\mathrm{AskH}_7] \\ &\Pr[\mathrm{Auth}_7] = \Pr[\mathrm{Auth}_6] = 0\end{aligned}$$

Thus, we have

$$\begin{aligned}\mathrm{Adv}_{\mathcal{P}}^{\mathrm{auth}}(\mathcal{A}) \le C' \cdot q_{send}^{s'} + 5 q_h \mathrm{Adv}_{\mathcal{P}}^{\mathrm{CDH}}(t + (q_{send} + q_{exe} \\ +1) \cdot \tau_e) + \frac{q_h^2 + 6q_{send}}{2^{l+1}} + \frac{(q_{send} + q_{exe})^2}{2p}. \quad \square\end{aligned}$$

References for Appendix

[1] F. T. B. Muhaya, "Cryptanalysis and security enhancement of zhu's authentication scheme for telecare medicine information system," *Secur. Commun. Netw.*, vol. 8, no. 2, pp. 149–158, 2015.

[2] L. Xu and F. Wu, "An improved and provable remote user authentication scheme based on elliptic curve cryptosystem with user anonymity," *Secur. Commun. Netw.*, vol. 8, no. 2, pp. 245–260, 2015.

[3] D. Mishra, A. K. Das, A. Chaturvedi, and S. Mukhopadhyay, "A secure password-based authentication and key agreement scheme using smart cards," *J. Inform. Secur. Appl.*, vol. 23, pp. 28–43, 2015.

[4] V. Odelu, A. K. Das, and A. Goswami, "An effective and robust secure remote user authenticated key agreement scheme using smart cards in wireless communication systems," *Wirel. Pers. Commun.*, vol. 84, no. 4, pp. 2571–2598, 2015.

[5] J. W. Byun, "Privacy preserving smartcard-based authentication system with provable security," *Securi. Commun. Netw.*, vol. 8, no. 17, pp. 3028–3044, 2015.

[6] D. Wang, N. Wang, P. Wang, and S. Qing, "Preserving privacy for free: Efficient and provably secure two-factor authentication scheme with user anonymity," *Info. Sci.*, vol. 321, pp. 162–178, 2015.

[7] G. M. Yang, D. S. Wong, H. X. Wang, and X. T. Deng, "Two-factor mutual authentication based on smart cards and passwords," *J. Comput. Syst. Sci.*, vol. 74, no. 7, pp. 1160–1172, 2008.

[8] S. H. Wu, Y. F. Zhu, and Q. Pu, "Robust smart-cards-based user authentication scheme with user anonymity," *Secur. Commun. Netw.*, vol. 5, no. 2, pp. 236–248, 2012.

[9] I. Liao, C. Lee, and M. Hwang, "A password authentication scheme over insecure networks," *J. Comput. Syst. Sci.*, vol. 72, no. 4, pp. 727–740, 2006.

[10] R. Madhusudhan and R. Mittal, "Dynamic id-based remote user password authentication schemes using smart cards: A review," *J. Netw. Comput. Appl.*, vol. 35, no. 4, pp. 1235–1248, 2012.

[11] M. L. Das, "Two-factor user authentication in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1086–1090, 2009.

[12] P. Gope and T. Hwang, "Lightweight and energy-efficient mutual authentication and key agreement scheme with user anonymity for secure communication in global mobility networks," *IEEE Syst. J.*, 2015, doi:10.1109/JSYST.2015.2416396.

[13] Y. Lu, L. Li, H. Peng, and Y. Yang, "Robust anonymous two-factor authenticated key agreement schemem for mobile client-server environment," *Secur. Commun. Netw.*, 2016, doi: 10.1002/sec.1419.

[14] Q. Xie, N. Dong, D. S. Wong, and B. Hu, "Cryptanalysis and security enhancement of a robust two-factor authentication and key agreement protocol," *Int. J. Commun. Syst.*, vol. 29, no. 3, pp. 478–487, 2016.

[15] S. Islam, "Design and analysis of an improved smartcard-based remote user password authentication scheme," *Int. J. Commun. Syst.*, vol. 29, no. 11, pp. 1708–1719, 2016.

[16] T.-T. Truong, M.-T. Tran, A.-D. Duong, and I. Echizen, "Chaotic chebyshev polynomials based remote user authentication scheme in client-server environment," in *Proc. SEC 2015*, pp. 479–494.

[17] B. Djellali, K. Belarbi, A. Chouarfia, and P. Lorenz, "User authentication scheme preserving anonymity for ubiquitous devices," *Secur. Commun. Netw.*, vol. 8, no. 17, pp. 3131–3141, 2015.

[18] D. Mishra, A. K. Das, A. Chaturvedi, and S. Mukhopadhyay, "A secure password-based authentication and key agreement scheme using smart cards," *J. Inform. Secur. Appl.*, vol. 23, pp. 28–43, 2015.

[19] F. Wu, L. Xu, S. Kumari, X. Li, and A. Alelaiwi, "A new authenticated key agreement scheme based on smart cards providing user anonymity with formal proof," *Secur. Commun. Netw.*, 2015, doi: 10.1002/sec.1305.

[20] S. A. Chaudhry, M. S. Farash, H. Naqvi, S. Kumari, and M. K. Khan, "An enhanced privacy preserving remote user authentication scheme with provable security," *Securi. Commun. Netw.*, vol. 8, no. 18, pp. 3782–3795, 2015.

[21] S. Kumari, M. K. Khan, and X. Li, "An improved remote user authentication scheme with key agreement," *Comput. & Electr. Eng.*, vol. 40, no. 6, pp. 97–112, 2014.

[22] B. Chen and W. Kuo, "Robust smart-card-based remote user password authentication scheme," *Int. J. Commun. Syst.*, vol. 27, no. 2, pp. 377–389, 2014.

[23] J.-L. Tsai, N.-W. Lo, and T.-C. Wu, "Novel anonymous authentication scheme using smart cards," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2004–2013, 2013.

[24] X. Li, J. Niu, M. K. Khan, and J. Liao, "An enhanced smart card based remote user password authentication scheme," *J. Netw. Comput. Appl.*, vol. 36, no. 5, pp. 1365–1371, 2013.

[25] S. Kumari and M. K. Khan, "Cryptanalysis and improvement of 'a robust smart-card-based remote user password authentication scheme'," *Int. J. Commun. Syst.*, vol. 27, no. 12, pp. 3939–3955, 2014.

[26] D.-Z. Sun and Z.-F. Cao, "On the privacy of khan et al.'s dynamic id-based remote authentication scheme with user anonymity," *Cryptologia*, vol. 37, no. 4, pp. 345–355, 2013.

[27] T.-F. Lee and C.-M. Liu, "A secure smart-card based authentication and key agreement scheme for telecare medicine information systems," *J. Med. Syst.*, vol. 37, no. 3, 2013.

[28] S. H. Islam and G. Biswas, "Design of improved password authentication and update scheme based on elliptic curve cryptography," *Math. Comput. Model.*, vol. 57, no. 11-12, pp. 2703–2717, 2013.

[29] X. Li and Y. Zhang, "A simple and robust anonymous two-factor authenticated key exchange protocol," *Secur. Commun. Netw.*, vol. 6, no. 6, pp. 711–722, 2013.

[30] Y.-F. Chang, W.-L. Tai, and H.-C. Chang, "Untraceable dynamic-identity-based remote user authentication scheme with verifiable password update," *Int. J. Commun. Syst.*, vol. 27, no. 11, pp. 3430–3440, 2014.

[31] C.-T. Li, "A new password authentication and user anonymity scheme based on elliptic curve cryptography and smart card," *IET Inform. Secur.*, vol. 7, no. 1, pp. 3–10, 2013.

[32] K.-K. Kim and M.-H. Kim, "An enhanced anonymous authentication and key exchange scheme using smartcard," in *Proc. ICISC 2012*, ser. LNCS, A. Juels and C. Paar, Eds. Springer, vol. 7839, pp. 487–494,.

[33] R. Ramasamy and A. P. Muniyandi, "An efficient password authentication scheme for smart card." *Int. J. Netw. Secur.*, vol. 14, no. 3, pp. 180–186, 2012.

[34] Z. Zhu, "An efficient authentication scheme for telecare medicine information systems," *J. Med. Syst.*, vol. 36, no. 6, pp. 3833–3838, 2012.

[35] D. Wang and C. G. Ma, "Cryptanalysis and security enhancement of a remote user authentication scheme using smart cards," *Elsevier J. China Uni. Posts Telecommun.*, vol. 19, no. 5, pp. 104–114, 2012.

[36] W. Hsieh and J. Leu, "Exploiting hash functions to intensify the remote user authentication scheme," *Comput. Secur.*, vol. 31, no. 6, pp. 791–798, 2012.

[37] F. Wen and X. Li, "An improved dynamic id-based remote user authentication with key agreement scheme," *Comput. & Electr. Eng.*, vol. 38, no. 2, pp. 381–387, 2012.

[38] D. Wang, C. G. Ma, and P. Wu, "Secure password-based remote user authentication scheme with non-tamper resistant smart cards," in *Proc. DBSec 2012*, ser. LNCS. Springer, 2012, vol. 7371, pp. 114–121.

[39] D. He, J. Chen, and J. Hu, "Improvement on a smart card based password authentication scheme," *J. Internet Tech.*, vol. 13, no. 3, pp. 38–42, 2012.

[40] R. C. Wang, W. S. Juang, and C. L. Lei, "Robust authentication and key agreement scheme preserving the privacy of secret key," *Comput. Commun.*, vol. 34, no. 3, pp. 274–280, 2011.

[41] T. Chen, H. Hsiang, and W. Shih, "Security enhancement on an improvement on two remote user authentication schemes using smart cards," *Future. Gener. Comput. Syst.*, vol. 27, no. 4, pp. 377–380, 2011.

[42] M. Khan, S. Kim, and K. Alghathbar, "Cryptanalysis and security enhancement of a more efficient & secure dynamic id-based remote user authentication scheme'," *Comput. Commun.*, vol. 34, no. 3, pp. 305–309, 2011.

[43] J. Kim, H. Choi, and J. Copeland, "Further improved remote user authentication scheme," *IEICE Trans. Fund. Electron. Comm. Comput. Sci.*, vol. 94, no. 6, pp. 1426–1433, 2011.

[44] A. K. Awasthi, K. Srivastava, and R. Mittal, "An improved timestamp-based remote user authentication scheme," *Comput. & Electr. Eng.*, vol. 37, no. 6, pp. 869–874, 2011.

[45] C. T. Li and C. Lee, "A robust remote user authentication scheme using smart card," *Info. Tech. Control*, vol. 40, no. 3, pp. 236–245, 2011.

[46] S. K. Sood, "Secure dynamic identity-based authentication scheme using smart cards," *Inform. Secur. J.*, vol. 20, no. 2, pp. 67–77, 2011.

[47] X. Li, W. Qiu, D. Zheng, K. F. Chen, and J. Li, "Anonymity enhancement on robust and efficient password-authenticated key agreement using smart cards," *IEEE Trans. Ind. Elec.*, vol. 57, no. 2, pp. 793–800, 2010.

[48] J. Tsai, T. Wu, and K. Tsai, "New dynamic id authentication scheme using smart cards," *Int. J. Commun. Syst.*, vol. 23, no. 12, pp. 1449–1462, 2010.

[49] R. Song, "Advanced smart card based password authentication protocol," *Comput. Stand. & Inter.*, vol. 32, no. 5, pp. 321–325, 2010.

[50] S. Sood and K. Sarje, K.and Singh, "An improvement of xu et al.'s authentication scheme using smart cards," in *Proc. Compute 2010*. ACM, Jan. 2010, pp. 1–5.

[51] K. H. Yeh, C. Su, and N. W. Lo, "Two robust remote user authentication protocols using smart cards," *J. Syst. Soft.*, vol. 83, no. 12, pp. 2556–2565, 2010.

[52] W. Horng, C. Lee, and J. Peng, "A secure remote authentication scheme preserving user anonymity with non-tamper resistant smart cards," *WSEAS Trans. Inform. Sci. Appl.*, vol. 7, no. 5, pp. 619–628, 2010.

[53] D. Sun, J. Huai, J. Sun, J. Li, and Z. Feng, "Improvements of juang et al.'s password-authenticated key agreement scheme using smart cards," *IEEE Trans. Ind. Elec.*, vol. 56, no. 6, pp. 2284–2291, 2009.

[54] H. C. Hsiang and W. K. Shih, "Weaknesses and improvements of the yoon-ryu-yoo remote user authentication scheme using smart cards," *Comput. Commun.*, vol. 32, no. 4, pp. 649–652, 2009.

[55] J. Xu, W. Zhu, and D. Feng, "An improved smart card based password authentication scheme with provable security," *Comput. Stand. & Inter.*, vol. 31, no. 4, pp. 723–728, 2009.

[56] S.-K. Kim and M. G. Chung, "More secure remote user authentication scheme," *Comput. Commun.*, vol. 32, no. 6, pp. 1018–1021, 2009.

[57] H. Chung, W. Ku, and M. Tsaur, "Weaknesses and improvement of wang et al.'s remote user password authentication scheme for resource-limited environments," *Comput. Stand. & Inter.*, vol. 31, no. 4, pp. 863–868, 2009.

[58] R. Ramasamy and A. P. Muniyandi, "New remote mutual authentication scheme using smart cards," *Trans. Data Privacy*, vol. 2, pp. 141–152, 2009.

[59] W.-S. Juang, S.-T. Chen, and H.-T. Liaw, "Robust and efficient password-authenticated key agreement using smart cards," *IEEE Trans. Ind. Elec.*, vol. 55, no. 6, pp. 2551–2556, 2008.

[60] T.-H. Chen and W.-B. Lee, "A new method for using hash functions to solve remote user authentication," *Comput. & Electr. Eng.*, vol. 34, no. 1, pp. 53–62, 2008.

[61] X. Wang, W. Zhang, J. Zhang, and M. Khan, "Cryptanalysis and improvement on two efficient remote user authentication scheme using smart cards," *Comput. Stand. & Inter.*, vol. 29, no. 5, pp. 507–512, 2007.

[62] R.-C. Wang, W.-S. Juang, and C.-L. Lei, "A simple and efficient key exchange scheme against the smart card loss problem," in *IEEE/IFIP EUC 2007*, ser. LNCS, vol. 4809, pp. 728–744.

[63] S. Lee, H. Kim, and K. Yoo, "Improvement of chien et al.'s remote user authentication scheme using smart cards," *Comput. Stand. & Inter.*, vol. 27, no. 2, pp. 181–183, 2005.

[64] C. Fan, Y. Chan, and Z. Zhang, "Robust remote authentication scheme with smart cards," *Comput. Secur.*, vol. 24, no. 8, pp. 619–628, 2005.

[65] R. Lu and Z. Cao, "Efficient remote user authentication scheme using smart card," *Comput. Netw.*, vol. 49, no. 4, pp. 535–540, 2005.

[66] E. Yoon, E. Ryu, and K. Yoo, "Further improvement of an efficient password based remote user authentication scheme using smart cards," *IEEE Trans. Comsum. Electr.*, vol. 50, no. 2, pp. 612–614, 2004.

[67] W.-C. Ku and S.-M. Chen, "Weaknesses and improvements of an efficient password based remote user authentication scheme using smart cards," *IEEE Trans. Comsum. Electr.*, vol. 50, no. 1, pp. 204–207, 2004.

[68] S.-T. Wu and B.-C. Chieu, "A user friendly remote authentication scheme with smart cards," *Comput. Secur.*, vol. 22, no. 6, pp. 547–550, 2003.

[69] A. Awasthi and S. Lal, "A remote user authentication scheme using smart cards with forward secrecy," *IEEE Trans. Comsum. Electr.*, vol. 49, no. 4, pp. 1246–1248, 2003.

[70] H.-M. Sun, "An efficient remote use authentication scheme using smart cards," *IEEE Trans. Consum. Electr.*, vol. 46, no. 4, pp. 958–961, 2000.

[71] M.-S. Hwang and L.-H. Li, "A new remote user authentication scheme using smart cards," *IEEE Trans. Consum. Electr.*, vol. 46, no. 1, pp. 28–30, 2000.

[72] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment," *IEEE Trans. Depend. Secur. Comput.*, vol. 12, no. 4, pp. 428–442, 2015.

[73] D. Wang and P. Wang, "Offline dictionary attack on password authentication schemes using smart cards," in *Proc. ISC 2013*, ser. LNCS, Y. Desmedt, Ed., vol. 7807, pp. 221–237.

[74] C.-G. Ma, D. Wang, and S.-D. Zhao, "Security flaws in two improved remote user authentication schemes using smart cards," *Int. J. Commun. Syst.*, vol. 27, no. 10, pp. 2215–2227, 2014.

[75] D. Wang, C. G. Ma, S. Zhao, and C. Zhou, "Cryptanalysis of two dynamic id-based remote user authentication schemes for multi-server architecture," in *Proc. NSS 2012*, ser. LNCS, vol. 7645, pp. 462–475.

[76] D. Wang, P. Wang, and J. Liu, "Improved privacy-preserving authentication scheme for roaming service in mobile networks," in *Proc. IEEE WCNC 2014*, pp. 3178–3183.

[77] D. Wang and C. Ma, "Cryptanalysis of a remote user authentication scheme for mobile client-server environment based on ECC," *Inform. Fusion,*, vol. 14, no. 4, pp. 498–503, 2013.

[78] D. Wang and P. Wang, "On the anonymity of two-factor authentication schemes for wireless sensor networks: Attacks, principle and solutions," *Comput. Netw.*, vol. 73, pp. 41–57, 2014.

[79] ——, "Understanding security failures of two-factor authentication schemes for real-time applications in hierarchical wireless sensor networks," *Ad Hoc Netw.*, vol. 20, pp. 1–15, 2014.

[80] K. Xue, P. Hong, and C. Ma, "A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture," *J. Comput. Syst. Sci.*, vol. 80, no. 1, pp. 195–206, 2014.

[81] V. Odelu, A. Das, and A. Goswami, "A secure biometrics-based multi-server authentication protocol using smart cards," *IEEE Trans. Inform. Foren. Secur.*, vol. 10, no. 9, pp. 1953–1966, 2015.

[82] J. Yu, G. Wang, Y. Mu, and W. Gao, "An efficient generic framework for three-factor authentication with provably secure instantiation," *IEEE Trans. Inform. Foren. Secur.*, vol. 9, no. 12, pp. 2302–2313, 2014.

[83] J. Bonneau, M. Just, and G. Matthews, "Whats in a name?" in *Proc. FC 2010*, ser. LNCS, R. Sion, Ed. Springer, 2010, vol. 6052, pp. 98–113.

[84] J. Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Proc. IEEE S&P 2012*, pp. 538–552.

[85] "Amazon elastic compute cloud (Amazon EC2)," Amazon Web Services Inc., 2015, https://aws.amazon.com/ec2/pricing/.

[86] D. Wang and P. Wang, "On the implications of Zipf's law in passwords," in *Proc. ESORICS 2016*, ser. LNCS, vol. 9878. Springer, pp. 1–21.

[87] M. Abdalla, F. Benhamouda, and P. MacKenzie, "Security of the j-pake password-authenticated key exchange protocol," in *Proc. IEEE S&P 2015*. IEEE Computer Society, 2015, pp. 571–587.

[88] J. Katz, R. Ostrovsky, and M. Yung, "Efficient and secure authenticated key exchange using weak passwords," *J. ACM*, vol. 57, no. 1, pp. 1–41, 2009.

[89] X. Yi, F. Hao, L. Chen, and J. Liu, "Practical threshold password-authenticated secret sharing protocol," in *Proc. ESORICS 2015*, ser. LNCS, vol. 9326, pp. 347–365.

[90] M. Shirvanian, S. Jarecki, N. Saxena, and N. Nathan, "Two-factor authentication resilient to server compromise using mix-bandwidth devices," in *Proc. NDSS 2014*. The Internet Society, pp. 1–16.

[91] N. Fazio, R. Gennaro, I. M. Perera, and W. E. Skeith III, "Hard-core predicates for a diffie-hellman problem over finite fields," in *Proc. CRYPTO 2013*, pp. 148–165.