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

Ding Wang Student Member, IEEE, 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 “passwords” with a “fuzzy-verifier”, our scheme hits “two birds”: it eliminates the long-standing security-usability conflict that is considered intractable in the literature, while achieving security guarantees beyond the conventional optimal security bound.

Keywords—Two-factor authentication, Smart card loss attack, Evaluation metric, Zipf’s law, Provable security.

I. INTRODUCTION

With the rapid development of distributed systems and the increasing demand for sharing services and resources, secure and efficient communications between the distributed user terminals and service providers are raising more and more concerns, and it is of utter importance to protect the systems (e.g., e-commerce systems [1] and cloud systems [2]) and the users’ privacy and security from malicious adversaries. Accordingly, user authentication becomes an essential security mechanism for application systems to ensure the authenticity of the communicating parties. Among the numerous methods for user authentication, password authentication is the most widely used and acceptable mechanism because of its easy-operation, scalability, compatibility and low-cost advantages [3], [4]. In such password-only authentication schemes (some notable ones include SRP [5], KOY [6] and J-PAKE [7]), each user is assumed to hold a memorable, low-entropy password, while the authentication server needs to store a password-related verifier table necessary to verify the authenticity of users.

An inherent limitation of the traditional password-only mechanism is that, the server has to store a sensitive verifier table that contains the passwords (or passwords in salted hash) of all the registered users. Once the authentication server is compromised, all the users’ passwords will be exposed. Due to human-beings’ inherently limited memory and security budget, the distribution of user-chosen passwords are highly skewed [8]. As a result, even if passwords are stored in salted-hash, this poses no real obstacle for an attacker to recover them by an overwhelming percentage (see [9]) by using modern probabilistic cracking techniques [10] and common hardware like GPUs. At Password’12, Gosney [11] 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 [12], but with the cost of an honest server increasing by the same factor with the attacker, while the later is likely to be better equipped with dedicated password-cracking hardware and software.

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 including Adobe (150M), Evernote (50M), LivingSocial (50M), Anthem (40M) [13], Rockyou (32M) [14], Tianya (30M), Dodonew (16M), CSDN (6.4M) [15], Gmail (4.9M), Forbes (1.1M) [16] and Phpbb (255K), just to name a few. Some services (e.g., Anthem [13] and Phpbb [17]) even have been breached more than once during the last five years. What makes things worse is that, users tend to utilize the same password (or slight variations) to access multiple servers (e.g., 43-51% of users [18]), a breach of one server will lead to the failure of all other servers, which is described as the “domino effect” of password re-use.

To deal with the issue of password leakage from a compromised server, threshold password-only authentication schemes (e.g., [19]) have recently been suggested. In such schemes, the password files and user data are distributed over multiple servers, and thus no coalition of servers up to a certain threshold can learn anything about the password. However, they are inherently unable to address another emerging issue—password leakage at the user side (e.g., hidden camera, key-loggers and phishing). To prevent user side password leakage, leakage-resilient password systems (LRPS) [20] have been proposed. In such schemes, a user needs to input the password indirectly and this imposes an extra burden on common users. Recently, it has been revealed [21] that, to attain both reasonable security and acceptable usability, LRPS schemes have to employ certain trusted devices to well address the threat of password leakage on user side.

The limitations of threshold and LRPS schemes entrench the third approach—introducing smart cards as “a second line of defense”. Smart-card-based password authentication schemes have been suggested as early as
about thirty years [22], and this kind of authentication is often termed as “two-factor authentication”. It has been widely deployed for various kinds of security-critical applications, such as e-commerce, e-banking and e-health, and also constitutes the basis of three-factor authentication [23], [24]. The participants of this sort of authentication (see Fig. 1) mainly involve a client $U$ and an authentication server $S$ [25], [26]. At first, the user $U$ registers to $S$ by submitting her self-chosen personal data (e.g., her identity and password) to $S$, then $S$ securely issues $U$ a smart-card with some security parameters. This phase is called 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 (SMS) 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” [25], 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 [27], [28], 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 [29], [30] 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 [26], [31]–[34]). However, in most of these 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 do not attempt (or 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 proposed to be secure turn out to be vulnerable. The research history of this area falls into the unsatisfactory cycle:

NEW PROTOCOL $\rightarrow$ BROKEN $\rightarrow$ IMPROVED PROTOCOL $\rightarrow$ BROKEN AGAIN $\rightarrow$ FURTHER IMPROVED PROTOCOL $\rightarrow$ $\cdots$ 

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.

### A. 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 [25] 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 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 [27]. Unfortunately, Wang et al. [25] 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 password-based scheme (e.g., see some notable schemes in [26], [30], [34], [35]), 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 [26], [30], [34], [36] 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 $\epsilon$ denotes a negligible value. However, user-chosen passwords are far from uniformly distributed and actually, they favor the other extreme: as shown in Sec. V-C, on average $\mathcal{A}$ would gain an advantage of 3.33%, 4.28%, 8.21%, 15.09% in just 3, 10, 10² and 10³ online guessing attempts, respectively; but not the assumed advantage of $3/10^9$, $10/10^9$, $10^2/10^9$, $10^3/10^9$, respectively. This misconception about the realistic security guarantee is evidently underestimated — the actual level of security risk of $\mathcal{P}$ is largely underestimated.

\footnote{Note that the size of user password space $\mathcal{D}$ increases as the user base increases, and thus it is not a constant (see Sec. V-C). It is generally assumed to be about $2^{30} \approx 10^9$ [37] as a rule of thumb.}
B. 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 $q_{send}/|D| + \epsilon$.

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” [38], traditionally the purview of system security, into two-factor 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 addresses the seemingly intractable security-usability problem left in [27]—whether or not there exist secure smart-card-based password authentication schemes and the password-changing phase does not need any interaction with the server”?

3. Third, we demonstrate that the proposed 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 show 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 in this work).

II. ADVERSARY MODEL AND EVALUATION CRITERIA

To fill the gap in fairly assessing a two-factor authentication scheme, in this Section we explicitly define an adversary model consistent with the reality 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 [3] for evaluating web authentication schemes based on usability and security principles.

A. Adversary model

In the conventional password authenticated key exchange (PAKE) protocols, the attacker $A$ is modeled to have full control of the communication channel between the communicating parties [39], such as eavesdropping, intercepting, inserting, deleting, and modifying any transmitted messages over the public channel. To characterize forward secrecy, $A$ may also be allowed to corrupt valid parties to attain long-term keys. Besides, previous session key(s) may be obtained by $A$ because of a number of reasons such as 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 [40], [41], the software loophole exploiting attacks (launched on software-supported
card, e.g., Java Card) [42] or reverse engineering techniques [43], [44]. 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 [32], [45]) and impersonation attack (e.g., the problematic schemes in [35], [46]). 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 [28] 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 card reader and then reading the information stored in the card via the stolen (or lost) smart card, otherwise this combination will enable the attacker to trivially break any two-factor authentication protocols. This treatment adheres to “the extreme-adversary principle” [47]: 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 \textit{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 and picked by the attacker), 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 un-trusted 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.

Our above analysis also invalidates the overly conservative proposition [26], [48] 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-cards-based schemes are completely insecure when used in un-trusted terminals, all the schemes based on such an extreme assumption like that of [26], [48] are as insecure as memory-cards-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 [26], [48].

Furthermore, for the sake of user-friendliness, a user is often allowed to select her own identity \textit{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 \textit{A} within polynomial time. Hence, in practice, it is realistic to assume that \textit{A} can offline enumerate all the (\textit{ID}, \textit{PW}) pairs in the Cartesian product \(\mathcal{D}_{id} \times \mathcal{D}_{pw}\) within a reasonable amount of time, where \(\mathcal{D}_{pw}\) denote the password space and \(\mathcal{D}_{id}\) denote the identity space, respectively.

\begin{table}[h]
\centering
\caption{Capabilities of the adversary}
\begin{tabular}{|c|}
\hline
C-00 \textit{The adversary} \textit{A} can offline enumerate all the elements in the Cartesian product \(\mathcal{D}_{id} \times \mathcal{D}_{pw}\) within a reasonable amount of time, where \(\mathcal{D}_{pw}\) and \(\mathcal{D}_{id}\) denote the password space and the identity space, respectively. \\
C-01 \textit{The (active) adversary} \textit{A} has the capability of determining the victim’s identity. \\
C-1 \textit{The adversary} \textit{A} has full control of the communication channel between the communicating parties, such as eavesdropping, intercepting, inserting, deleting, and modifying any transmitted messages over the public channel. \\
C-2 \textit{The adversary} \textit{A} may either (1) learn the password of a victim via malicious card reader, or (ii) extract the secret data in the lost smart card by side-channel attacks, but cannot achieve both. Otherwise, it is a trivial case. \\
C-3 \textit{The adversary} \textit{A} can learn the previously established session key(s). \\
C-4 \textit{The adversary} \textit{A} can learn the server’s long-time private key(s) as well as all other stored data only when evaluating the eventual failure of the server. \\
\hline
\end{tabular}
\end{table}

The capabilities of \textit{A} in our model are summarized in Table I. As far as we know, our work, following the works in [26], [28], [50] while providing new insights, is one of the few ones that explicitly specify the capabilities of the adversary. 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 [28], Wang presented three kinds of security models, namely Type I, II and III. The last model is the most powerful one to date, and it mainly makes three assumptions:

1. \textit{A} is allowed to have full control of the communication channel, which is consistent with C-1 in Table I;
2. The smart card is assumed to be non-tamper resistant and the user’s password may be intercepted by \textit{A} using a malicious card reader, but not both, which is consistent with C-2 in Table I;
3. The smart card has no counter protection, i.e. \textit{A} can issue a large amount of queries to the card using a malicious card reader to learn some 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 \textit{A} can learn some useful information (except the static data stored in the card, which can be learnt by \textit{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, \textit{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), which is explicitly not allowed in the Type III model. Hence, Assumption 3 is not considered in our model. As with [49], 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 [28], and the key difference is that $A$ in Type III model is not provided with 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 [50] and Yang et al.’s model [26], our model has explicitly taken the malicious card reader into consideration, and $A$ is further armed with C-3 and C-4.

Moreover, $A$ in our model is assumed to be able to offline enumerate all the $(ID, PW)$ pairs in the Cartesian product $D_{id} \times P_{pw}$ 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., [46], [49]). Note that, C-01 has also been yet implicitly made in [26], [28], both of which do not concern the admired feature of user anonymity, for the emphasis on C-01 (e.g., we deliberately separate it from C-00 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 (i.e., the computational power of $A$ is large but not omnipotent) assumptions, 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 build (see Sec. IV).

B. Evaluation criteria

As pointed out by Yang et al. [26], 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 [29] 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 [25]) among the criteria.

Considering these earlier criteria-related studies [25], [26], [29], based on our cryptanalysis experience of 67 typical schemes (see Appendix A and some of our earlier attacking results [25], [49], [51]) and further using the iteration methodology [3] of criterion refinement, we put forward a broads list of 12 independent criteria in terms of user friendliness and security that a two-factor authentication scheme shall satisfy:

C1. No verifier-table: the server does not need to maintain a database for storing the passwords or some derived values of the passwords of its clients;

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, guess the password of the user by using 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 can resist various kinds of basic and 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 the smart 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 $A$ has obtained access to the victim’s smart card, while C5 deals with scenarios where $A$ has no access to the victim’s smart card. The criterion C4 considers the traditional smart-card-loss issues (see [29]) as well as attacking scenarios newly revealed (e.g., attacks by interactively using the server as an oracle [27] and attacks by returning the extracted card [25]); C5 is based on the list of basic attacks [52] that a password-only authentication scheme needs to guard against and on security notions [53] that relate to session keys, 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 [29], 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”, for 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 [26], [30], is missing in [29]. One can check that the criteria in [29] 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. [54], the criterion concerning with performance, which says “The scheme must be efficient and practical”, is not

\footnote{As pointed out to us by Prof. Michael Scott, authentication protocols (no matter password-only or two-factor) in which the server also serves as the registration center of clients are inherently unable to resist key compromise impersonation (KCI) attack, and therefore the KCI notion is not considered.}
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 [30], [54] are included into our set. Although the criterion related with performance is not listed in Yang et al.'s set [26], their set is merely composed of four criteria (i.e., C2-C5) and evidently too limited to be of practical operability.

Finally, as one could argue that there may be the probability that someone else will claim tomorrow that a list they come up with is better, we acknowledge that our list of criteria is not complete, and indeed, any such list could always be expanded. We resist the temptation to create a new criterion and mainly focus on the criteria that have already been extensively discussed in the past literature, because a criterion that has drawn little or no previous attention probably can not be considered essential. Though not cast in stone, our list is more concrete and comprehensive than related ones and is not complete, and indeed, any such list could always be expanded. We resist the temptation to create a new criterion and evidently too limited to be of practical operability.

To formally capture the capabilities of an adversary in smart-card-based password authentication and specify how the concrete, concise and comprehensive as compared to related model is the harshest one so far and our criterion set is more expected to help facilitate a deeper understanding of the pros and cons of the current and future two-factor schemes. This is of fundamental importance for security engineers to make their choices correctly and for protocol designers to develop practical schemes with better usability-security tradeoffs.

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 related works. As any cryptographic protocol meets its goals only within some security model [25], [39], 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 demonstrated and tested by rating 67 two-factor schemes without hidden agenda, as summarized in a carefully constructed table in Appendix A. Both the rating criteria and their definitions were iteratively refined and re-categorized when evaluating these 67 schemes. One can see that, each criterion can be satisfied by at least 15 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.

III. 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 [39] 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 refined version proposed by Bresson et al. [55] 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 [39], [55] for more details.

Players. In a two-factor protocol \( \mathcal{P} \), there are two protocol participants involved, namely, a user \( U \in \text{User} \) and a server \( S \in \text{Server} \), 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 \( I \in \text{User} \cup \text{Server} \).

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 \((\mathit{pub}_{\text{key}}, \mathit{pri}_{\text{key}})\) in a public-key based scheme (or a single symmetric key \( \mathit{sym}_{\text{key}} \) in a symmetric-key based scheme), while each user \( U \in \text{User} \) with identity \( ID_U \) is equipped with a password \( PW_U \) which is assumed to be drawn from a Zipf-distributed [8] “dictionary” \( \mathcal{D} \) of small size \( |\mathcal{D}| \), where \( |\mathcal{D}| \) is a fixed constant which is independent of the system security parameter. The vector \( < ID_U, \mathit{PW}_U > \in \text{User} \) is kept on \( S \), where \( \mathit{TPW}_U \) is an injective transformation of \( < ID_U, \mathit{PW}_U > \) and \( \mathit{pub}_{\text{key}}/\mathit{sym}_{\text{key}} \). 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 \( A \) and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. The query types available to \( A \) are defined as follows.

- \( \text{Execute}(U^i, S^j) \): 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.

- \( \text{Send}(I, m) \): This query models an active attack, in which \( A \) 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 \( \text{Send}(U^i) \) (Start) initializes the protocol. Start is a message, and thus \( A \) receives the flow that the client should send out to \( S \).

- \( \text{Test}(I) \): This oracle query is not used to simulate the adversary’s attack, but to define session key’s semantic security. If no session key for instance \( I \) is defined, then undefined symbol \( \bot \) is returned. Otherwise, a private coin \( c \) is flipped. If \( c = 1 \) then the session key \( sk \) is returned to \( A \), 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.

- \( \text{Reveal}(I, a) \): This query models the corruption capability of \( A \) (i.e. C-2 and C-4 in Table I). 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 \( \bot \) is returned.

- \( \text{Corrupt}(I, a) \): This query models corruption capability of \( A \) (i.e. C-2 and C-4 in Table I). \( A \) can break either one of the two authentication factors of clients, 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.

When performing security proofs, most existing password-based protocols (e.g., [6], [7], [26], [34], [39]) assume passwords to be uniformly distributed with no qualities of conscience. Some prudent ones (e.g., [6], [39]) may further argue that, if user passwords are not uniformly distributed, one can easily adjust the calculation of \( A \)'s advantages. However, human-chosen passwords are actually extremely skewed (i.e., Zipf-distributed [8]) and as will be shown in Fig. 3 and Sec. V, there are three to four order-of-magnitude difference in \( A \)'s advantages between these two password distributions. This defeats the purpose of using formal methods to accurately assess \( A \)'s advantages. Thus, the uniform assumption of passwords is undesirable and shall be abandoned.
• If \( I = S, a = 1 \), it outputs the private key \( pr_{i\text{key}} \) (or the symmetric key \( pr_{i\text{key}} \)) of the server \( S \) and \( <ID_U, TPW_U > \in User \) 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 listed in Table 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.

**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 [6], 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 the adversary. 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 \( I \) or its partner 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 the adversary from impersonating the client or the server. We denote by \( Adv_{\text{auth}}^\mathcal{P}(A) \) the probability that \( A \) 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 to be 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 [6, 39, 52], the best strategy that an adversary 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 \( A \)’s best possible strategy to impersonate a party; (1) if the password is compromised but not the smart card, \( A \) 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, \( A \) 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 \( A \) making at most \( q_{\text{send}} \) on-line attacks, there exists a negligible function \( \epsilon(\cdot) \) such that:

\[ \text{Adv}_{\mathcal{P}}^\mathcal{A} = 2\Pr[\text{Succ}(\mathcal{A})] - 1 = 2\Pr|c' = c| - 1 \]

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

---

4An instance accepts when it has done sending and receiving messages and believes that it has established a session key with its intended partner.
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 \( A \) making at most \( q_{\text{rand}} \) on-line attacks, there exists a negligible function \( \epsilon(\cdot) \) such that:

\[
\text{Adv}_{\mathcal{P},\mathcal{D}}^{\text{adv}}(A) \leq \frac{\sum_{j=1}^{q_{\text{rand}}} \frac{1}{\ell}}{\sum_{i=1}^{\ell} 1} + \epsilon(\ell)
\]

where the parameters are the same with those of the definition of “Authentication”. This result requires that calls to the Execute-query (i.e., passive eavesdropping) are useless to \( A \).

**IV. 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 Sec. II-B. Our scheme consists of four phases: the registration phase, the login phase, the verification phase and the password change phase. For ease of presentation, we employ some intuitive abbreviations and notations listed in Table II.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U_i )</td>
<td>2nd user</td>
</tr>
<tr>
<td>( S )</td>
<td>remote server</td>
</tr>
<tr>
<td>( A )</td>
<td>the adversary</td>
</tr>
<tr>
<td>( ID_i )</td>
<td>identity of user ( U_i )</td>
</tr>
<tr>
<td>( PW_i )</td>
<td>password of user ( U_i )</td>
</tr>
</tbody>
</table>

**A. Registration phase**

The protocol is defined over a finite cyclic group \( G = \{g\} \), of order a \( \ell \)-bit prime number \( q \). This group could be a finite field or an elliptic curve group. In this paper, we assume \( G \) is a prime order subgroup of \( \mathbb{Z}_q^* \), where \( \mathbb{Z}_q^* = \{1, 2, \ldots, p-1\} \) and \( p \) is a large prime number such that \( q | p - 1 \). Hash functions from \( \{0, 1\}^* \rightarrow \{0, 1\}^l \) are denoted by \( H_i(\cdot) \), where \( l \) 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 \( A_j \)). Let \( (x, y = g^x \mod p) \) 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.** \( U_i \rightarrow S : \{ID_i, H_0(b|PW_i)\} \).

**Step R3.** On receiving the registration message from \( U_i \) at time \( t \), \( S \) first picks a random number \( a_1 \) and computes \( A_1 = H_0((H_0(ID_i) \oplus H_0(b|PW_i)) \mod n_0) \). Then \( S \) checks whether \( U_i \) is a registered user. If it is \( U_i \)'s initial registration, \( S \) creates a new entry for \( U_i \) in the account-database and stores \( \{ID_i, T_{\text{reg}}=T, a_1, \text{Honey\_List}={\text{Null}}\} \) in this entry. Otherwise, \( S \) only updates the values of \( T_{\text{reg}} \) to \( T, a_1 \) to newly created \( a_1 \), and \( \text{Honey\_List} \) to \( \text{Null} \) in the existing entry for \( U_i \). Next, \( S \) computes \( N_i = H_0(b|PW_i) \oplus H_0(x|ID_i) \mod T_{\text{reg}} \).

**Step R4.** \( S \rightarrow U_i : \) A smart card with security parameters \( \{N_i, A_1 \oplus a_1, q, g, y, n_0, H_0(\cdot), \ldots, H_3(\cdot)\} \).

Step R5. Upon receiving the smart card \( SC = \{C_1, C_2, C_3\} \), \( U_i \) activates 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 Sec. II-B). In addition, we assume, for simplicity, that the record \( T_{\text{reg}} \) 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 \( T_{\text{reg}} = T \parallel X \), where \( X \) is a large random number.

**B. 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 \( A_j = H_0((H_0(ID_i) \oplus H_0(b|PW_i)) \mod n_0) \) and verifies the validity of \( ID_i \) and \( PW_i \) by checking whether \( A_j \) equals the stored \( A_j \). If they are not equal, the session is terminated.

**Step L3.** \( SC \) chooses a random number \( u \) and computes \( C_1 = g^u \mod p \), \( Y_1 = g^y \mod p, k = H_0(x) \parallel ID_i \parallel T_{\text{reg}} = N_i \parallel H_0(b|PW_i) \parallel H_0(ID_i) \parallel \text{CAK}_i = (a_i|k) \parallel H_0(C_1|Y_1) \parallel M_i = H_0(Y_1|k||\text{CID}\|\text{CAK}_i) \).

**Step L4.** \( U_i \rightarrow S : \{C_1, \text{CID}_i, \text{CAK}_i, M_i\} \).

Note that, the verifer \( M_i \) in the login request is added to cope with the ability C-01 of the adversary, i.e., to resist against a new kind of offline password guessing attack on two-factor authentication uncovered by Huang et al. in 2014 [27].

**C. Verification phase**

After receiving the login request \( \{C_1, \text{CID}_i, \text{CAK}_i, M_i\} \), the server \( S \) performs the following operations:

**Step V1.** \( S \) computes \( Y_1 = (C_1)^x \mod p \) using its private key \( x \). Then, \( S \) derives \( D_i = \text{CID}_i \oplus H_0(C_1|Y_1) \) and checks whether \( D_i \) is in the correct format. If \( D_i \) is not valid, the session is terminated. Otherwise, \( S \) proceeds to the next step.

**Step V2.** \( S \) computes \( k = H_0(x||ID_i||T_{\text{reg}}) \) and \( M_i* = H_0(Y_1||k||\text{CID}_i||\text{CAK}_i) \), where \( T_{\text{reg}} \) 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 \( a_i^* k^i = \text{CAK}_i \oplus H_0(C_1|Y_1) \), and checks whether \( a_i^* \) equals the stored \( a_i \). An equality implies a login request with the right \( A_j \), \( S \) rejects if they are not equal. Otherwise, \( S \) checks whether the derived \( k^i \) 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 \( k^i \neq k \), implying that there is a 1/2\(^n_0\) probability that \( U_i \)'s card has been corrupted. Accordingly, \( S \) performs either (1) inserts \( k^i \) into \( \text{Honey\_List} \) when there are less than \( n_0 \) (e.g., \( n_0 = 10 \)) items in \( \text{Honey\_List} \); or (2) suspends \( U_i \)'s card (i.e., when there are \( n_0 \) items in \( \text{Honey\_List} \)) until \( U_i \) re-registers.

**Step V4.** \( S \) generates a random number \( v \) and computes the temporary key \( K_S = (C_1)^v \parallel H_0(C_2) \parallel H_3 = H_1(ID_i||ID_S|||Y_1||C_2||K||K_S) \).
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 \( K_U = (C_2)^u \mod r \),
\( C_4^* = H_2(\|ID_i\|\|ID_S\|Y_i\|C_2\|k\|K_U) \), and compares \( C_4^* \) with the received \( C_4 \). This equivalency authenticates
the legitimacy of the server \( S \), and \( U_i \) goes on to compute \( C_4 = H_2(\|ID_i\|\|ID_S\|Y_i\|C_2\|k\|K_U) \).
Step V7. \( U_i \rightarrow S : \{C_4\} \).
Step V8. Upon receiving \( \{C_4\} \) from \( U_i \), \( S \) first computes
\( C_1 = H_2(\|ID_i\|\|ID_S\|Y_i\|C_2\|k\|K_S) \) and then checks if
\( C_1^* \) equals the received \( C_1 \). If this verification holds, \( S \) authenticates the user \( U_i \) and the login
request is accepted else the connection is terminated.
Step V9. The user \( U_i \) and the server \( S \) agree on the common
session key \( sk_{U_i} = H_3(\|ID_i\|\|ID_S\|Y_i\|C_2\|k\|K_U) =
H_3(\|ID_i\|\|ID_S\|Y_i\|C_2\|k\|K_S) = sk_{S} \) for securing
data future communications.

D. 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 \( A_i^* = H_0(\|ID_i\|\oplus H_0(b||
PW_i)) \mod n_0 \) and verifies the validity of \( A_i^* \) by
checking whether \( A_i^* \) equals the stored \( A_i \). If the
verification holds, it implies the input \( ID_i \) and \( PW_i \)
are valid with a probability of \( \frac{n_0}{2n_0} \approx \left(\frac{0.01}{2}\right) \),
otherwise, the smart card rejects.
Step P3. The smart card asks the cardholder to resubmit a
new password \( PW_i^{new} \) and computes \( N_i^{new} =
N_i \oplus H_0(b||PW_i) \oplus H_0(b||PW_i^{new}) \), \( A_i^{new} =
H_0((\|ID_i\|\oplus H_0(b||PW_i^{new} \oplus d) \mod n_0) \). 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.

V. 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.

A. Basic ideas

To achieve the most essential goal — “truly two-factor
security”, a password-protected cryptographically strong
long-term secret \( k = H_0(x||ID_i||T_{reg}) \) is kept on the smart
card, where \( x \) is the server’s secret key. On the one hand, this
long-term secret \( k \) can be derived by the server 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 the server. 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 \( k = N_i \oplus H_0(b||PW_i^{*}) \),
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 \( A_1 = H_0(\|ID_i\|\|PW_i\|) \)
is stored in the card. Whenever \( U_i \) wants to change her password,
firstly she must submit her identity \( ID_i \) and password
\( PW_i^{*} \), then the card checks if \( H_0(\|ID_i\|\|PW_i\|) \) equals
the stored \( A_1 \). One can easily find that \( A \) can exhaustively search
the correct \( (ID_i, PW_i) \) pair in an offline manner once \( A_1 \) 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 [46], [49], [57], [58], where
the parameter \( A_1 \) is exactly computed in this insecure manner
and thus \( A \) can obtain the correct \( (ID_i, PW_i) \) pair once \( A_1 \)
has been revealed under the capability C-2(ii) in Table I.

However, if the parameter \( A_1 \) is computed as \( A_1 = H_0(\|ID_i\|
\oplus H_0(PW_i)) \mod n_0 \), one can be assured that there exists
\(|D_{id}| = 37 \) ≈ \( 2^{12} \) candidates of \((ID, PW) \) pair to frustrate
\( A \) when \(|D_{id}| = |D| \) = \( 10^6 \) [10], [37] and \( n_0 = 2^n \),
where \(|D_{id}| \) and \(|D| \) denote the size of the identity space and password space. Even specifically, with the capability of C-
01 (i.e., the victim user’s identity has already been learnt), \( A \)
will still be frustrated, because there exist \(|D|^{\frac{1}{n_0}} \approx \frac{2^0}{2^n} \) password candidates (as will be empirically established in Sec. V-B). To further exclude the specious passwords from the remaining \( 2^n \) 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” [38] (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 Sec. V-C,
we will demonstrate that this event can be timely detected
with an accuracy of \( 1 - \frac{1}{7} \). In Sec. VI, we will rigorously show
that this is the best strategy that \( A \) can exploit to obtain
the password and thereby to break the protocol. In this manner, we
thwart \( A \) from obtaining the correct \((ID, PW) \) pair and call
the parameter \( A_1 \) calculated through this new method “a fuzzy
verifier”. Note that, we do not directly store \( A_1 \) on the server
side, but instead store a random number \( a_i \) corresponding to
\( A_1 \) on \( S \) and also store both \( A_1 \) and \( a_i \oplus A_1 \) 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 \( A \) who gets temporary access
to \( U_i \)’s smart card may exploit this “security-usability trade-off
parameter” \( A_1 \). More specifically, \( A \) may attempt to change
\( U_i \)’s password to an arbitrary value and then returns the card
back, which constitutes a denial of service (DoS) attack. It is
difficult to see that, \( A \) will succeed with a probability only about \( \frac{1}{100} \) \( \approx \frac{0.0}{2} \), 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 \( H_0(\cdots) \) outputs randomly (which is
widely assumed [39]). This means \( A \) will not succeed easily
(i.e., with a chance of 50% after 26 days of attack). Even if this
TABLE III. GUESSING ENTROPY (GE) DISTRIBUTIONS OF $n_0 = 2^8$ PASSWORD POOLS. EACH POOL STEMS FROM THE DIVISION OF PASSWORD SPACE $D$ ACCORDING TO OUR FUZZY-VERIFIER $A_i$.

<table>
<thead>
<tr>
<th>Real-life password distributions</th>
<th>Percentage of pools with $GE &gt; 2^{12} \nu(2^{20} / n_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rockyou_top1Million</td>
<td>0.00%</td>
</tr>
<tr>
<td>Tianya_top1Million</td>
<td>10.16%</td>
</tr>
<tr>
<td>CSDN_top1Million</td>
<td>10.54%</td>
</tr>
<tr>
<td>Dodonew_top1Million</td>
<td>14.45%</td>
</tr>
<tr>
<td>Rockyou_top2Million</td>
<td>84.77%</td>
</tr>
<tr>
<td>Tianya_top2Million</td>
<td>96.09%</td>
</tr>
<tr>
<td>CSDN_top2Million</td>
<td>97.66%</td>
</tr>
<tr>
<td>Dodonew_top2Million</td>
<td>98.83%</td>
</tr>
<tr>
<td>Rockyou_topx2Million (x ≥ 3)</td>
<td>99.61%</td>
</tr>
<tr>
<td>Tianya_topx2Million (x ≥ 3)</td>
<td>100.00%</td>
</tr>
<tr>
<td>CSDN_topx2Million (x ≥ 3)</td>
<td>100.00%</td>
</tr>
<tr>
<td>Dodonew_topx2Million (x ≥ 3)</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

*For more details, readers can see http://wangdingg.weakely.com/fuzzyverifier.html.

DoS attack succeeds, $U_i$ can restore her card by re-registration. As discussed above and will be proved in Sec. VI, this DoS attack and the aforementioned online guessing attack are the two greatest threats that $A$ poses to our protocol. Fortunately, $A$ benefits little from this DoS attack and $A$’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 $g_{send} / |D| + \epsilon$).

To avoid password exposure (C3), $H_0(b || PW_i)$ instead of $PW_i$ or $h(PW_i)$ is submitted to server $S$, where $b$ is a random number unknown to the server $S$; to achieve $C6$, an entry $(D_i, T_{reg})$ 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 $CID_i$; to achieve forward secrecy (C12), Diffie-Hellman key exchange technique is adopted; and C4 and C10 will be further rigorously proved in Sec. VI.

B. Effectiveness of “fuzzy-verifier”

We now use large-scale real-life passwords to show that our proposed “fuzzy-verifier” $A_i = H_0((H_0(ID_i) \oplus H_0(b || PW_i)) \mod n_0)$ is effective in dividing $A$’s password guessing space, leaving adequate candidates for $A$ to identify and thus making it possible for “honeywords”, as will be shown in Sec. V-C, to bound $A$’s advantage to a low value.

Assume $A$ has obtained $U_i$’s card and extracted the parameter $A_i$. With the knowledge of $A_i$, $A$ can reduce the size of her guessing space $D$ to $\frac{|D|}{n_0}$. We proceed to show that $D/n_0$ is practically large enough. It is natural for us to approximate $D$ using real-life password accounts with frequency. As $A$ is clever, when performing an online guessing, she would always try the most likely password candidate first [37], and this attacking strategy is best approximated by the security notion of guessing entropy (GE):

$$G(D) = \sum_{i \in D} p_i \cdot i$$

where, without loss of generality, each password $pw_i$ in $D$ is assumed to be associated with a probability $p_i$ and $p_1 \geq p_2 \geq p_3 \geq \cdots$. It is not difficult to see that $G(D)$ is the expected number of guesses required to find the correct password $PW_i$.

Here we employ four datasets: 32 million Rockyou [14], 30 million Tianya, 16 million Dodonew and 6 million CSDN [15]. 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 $A$’s guessing space $D$, but use the most vulnerable distributions (i.e., portions with the top popular passwords). The underlying reason is that: (1) $A$ 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, these less vulnerable portions would naturally reach the goal.

Table III shows that, when the size of the dataset is no less than 3M, each divided pool indeed can reach a GE larger than $2^{12}$ (when $n_0 = 2^8$). This suggests that our “fuzzy-verifier” is indeed effective for these services with a user-base no less than 3M, 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) to Dodonew, their guessing space shall be as large as 2M to reach a GE≥$2^{12}$, while passwords created under a context similar to Rockyou shall be with a space of 3M.

C. Effectiveness of “fuzzy-verifier” + “honeywords”

In our scheme, to provide the admired property of “local and secure password change” [25], $U_i$ stores the “fuzzy-verifier” $A_i = H_0((H_0(ID_i) \oplus H_0(b || PW_i)) \mod n_0)$ in its card memory. This also facilitates $A$ to reduce her password guessing space size from $|D|$ to $|D| / n_0$ in an offline manner. In the above section, we have shown that our “fuzzy-verifier” is effective in making $|D| / n_0$ 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, $A$ is still prevented from gaining a large advantage in determining the final correct password $PW_i$ from the $|D| / n_0$ 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 $1 - \frac{1}{|D|^{m_0}}$ after just 10 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 event occurs $m_0$ times, then $S$ is with a confidence $1 - (\frac{1}{n_0})^{m_0}$ (e.g., which equals $1 - \frac{1}{2^{30}}$ 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. $P_1[Err\_change \leq \frac{1}{n_0}]$ shall be as small as possible, where $Err\_change$ denotes the event that, $U_i$ accidentally types an unintended password $PW_i$ when changing password, yet $H_0((H_0(ID_i) \oplus H_0(b || PW_i)) \mod n_0)$ equals $A_i$ and $PW_i$ is unwittingly changed to $PW_i^*$. R2. $P_2[Err\_detect \leq \frac{1}{n_0}]$ 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. \( Pr[\text{Succ}_\text{Ext}] = \frac{\sum_{j=1}^{m_0} \frac{1}{p_i}}{\sum_{j=1}^{D} \frac{1}{p_i}} \) shall be as small as possible, where \( \text{Succ}_\text{Ext} \) denotes the event that \( \text{Ext} \) occurs and \( A \) 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) \( Pr[\text{Err}_\text{detect}] \) decreases exponentially with \( m_0 \), while \( Pr[\text{Succ}_\text{Ext}] \) increases linearly with \( m_0 \), which means we can timely (and accurately) detect the event \( \text{Ext} \) and, at the meantime, confine \( A \)'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 \( m_0 = 2^8 \) and \( n_0 = 10 \). 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, \( Pr[\text{Err}_\text{detect}] \) shall be negligible and when \( m_0 \geq 10 \),

\[
Pr[\text{Err}_\text{detect}] = \frac{1}{(n_0 - m_0)} \leq \frac{1}{(2^8 - 10)}
\]

This indicates the event \( \text{Ext} \) can be detected with an accuracy of \( 1 - \frac{1}{10} \). On the other hand, when we set \( m_0 \leq 10 \) (and use the distribution of 16 million Dodonew passwords [15] for a concrete example),

\[
Pr[\text{Succ}_\text{Ext}] = \frac{\sum_{j=1}^{m_0} \frac{1}{p_i}}{\sum_{j=1}^{D} \frac{1}{p_i}} \approx \frac{\sum_{j=1}^{m_0} p_i}{\sum_{j=1}^{D} p_i} = 3.28\%
\]

We summarize in Table IV the results for 11 real-life password distributions with the guessing number (i.e., \( m_0 \)) varying from 1, 3, 10, 10^2 to 10^3, as well as varying from 1/10^4 percent to 1/5 percent. Thus, 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 IV is that, with just a handful of online guessing attempts, \( A \) can obtain a considerable amount of advantage. For instance, the average advantage of \( A \) will be up to 8.21% by only performing 100 online guesses (in an optimal order). This is in vast different from the traditional theoretical optimal security bound (see [30], [34], [36]): \( q_{\text{send}}/|D| + \epsilon \), where \( D \) is assumed to be uniformly distributed, and \( q_{\text{send}} \) denotes the number of online guessing that \( A \) engages in. Since \( D \) is generally assumed to be \( 10^8 \) [37] and \( q_{\text{send}} \) to be in thousands [56], \( q_{\text{send}}/|D| \) will be theoretically about 0.1%. Yet, the actual value is over 15.09%, which is far from a negligible risk level. From the last two rows in Table IV, one can see that there are two to four orders of magnitude difference in \( A \)'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 [59] and the fact that user passwords become weaker when additional authentication factor are in place [60], this gap will be even larger.

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

In a nutshell, before \( A \) sets off an alarm of user card corruption with a probability \( \frac{1}{(m_0 - 1)} \) 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 Sec. VI), while in conventional schemes this figure would generally be in hundreds or even in thousands. In this light, our scheme achieves security (i.e., \( Pr[\text{Succ}_\text{Ext}] \approx \frac{\sum_{j=1}^{m_0} p_i}{\sum_{j=1}^{D} p_i} \leq \sum_{i=1}^{10} p_i \) beyond the conventional optimal security bound (i.e., \( q_{\text{send}}/|D| + \epsilon \), serving as a hedge against human-beings’ limited memory.

### D. 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 schemes for both single-server architecture [26] and multi-server environment [61]. More specifically, a fuzzy-verifier (like our \( A_r \)) 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 \( \text{Honey List} \)) 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 \( A \) is exploiting the fuzzy-verifier as a oracle to reduce password space and to thwart \( A \)'s malicious action in a timely manner (e.g., by rate limiting or locking the account).

In Appendix B, we use two typical schemes, i.e. Tsai et al.’s scheme [32] and Xue et al.’s scheme [58], as case studies to demonstrate 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. In addition, we also employ Odelu et al.’s scheme [61] to briefly show its applicability to three-factor schemes. As a result, all the schemes (e.g., [24], [26], [30], [48])
previously subject to the long-standing C2 vs. C4 dilemma now can be relieved by using our proposed approach.

**Summary.** Both theoretical and empirical results show the practicality of integrating “honeywords” [38] 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 by Huang et al. [23]: “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”?

VI. **Formal security analysis**

In the following, we show that our scheme is secure in the model defined in Sec. III 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 $\ell$, algorithm $\text{Init}$ first runs an algorithm $\mathcal{G}$ to generate a group $\mathbb{G}$ of prime order $q$, where $|\mathbb{G}| = \ell$. Next, $\text{Init}$ generates a generator $g$ for $\mathbb{G}$, four collision-resistant hash functions $H_i : \{0,1\}^* \rightarrow \{0,1\}^{|i|}$ ($i = \{0, 1, 2, 3\}$), 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.

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

**Computational Diffie–Hellman (CDH) Assumption.** Let $G$ be a finite cyclic group of prime order $q$ generated by an element $g$, where the operation is denoted multiplicatively. A $(t, \epsilon)$-CDH attacker in $G$ is a PPT machine $\Delta$ running in time $t$. Then

$$\text{Adv}_{\Delta}^{\text{CDH}}(\epsilon) = \Pr[g^x, g^y = g^{xy}] - \epsilon$$

where $\epsilon$ is the advantage over the random values $x$ and $y$. The CDH-Assumption states that $\text{Adv}_{\Delta}^{\text{CDH}}(\epsilon) = \epsilon$ for any $t/\epsilon$ not too large.

**Theorem 1:** Let $G$ be a representative group, $D$ be a password space from which each user password is drawn according to the Zipf’s law $[\ell, 0.753771]$ and $n_0$ be the “security-useability trade-off parameter”. Let $P$ be the proposed authentication scheme stated in Section IV. Let $\mathcal{A}$ be a PPT adversary against the semantic security within a time bound $t$, with $q_{\text{send}}(\leq m_0 = 10)$ Send-queries and $q_{\text{exe}}$ Execution-queries, and making less than $q_{\text{r}}$ random oracle queries. Then we have

$$\text{Adv}_{\mathcal{A}}^{\text{P,D}}(\mathcal{A}) = 2\text{Pr}[\text{Succ}_{\mathcal{C}}] - 1 + 2\text{Pr}[\text{Succ}_{\mathcal{D}}]$$

$$\leq \sum_{j=1}^{q_{\text{send}}} \frac{1}{|\mathcal{D}|} \frac{q_{\text{r}}}{n_0} + \frac{2q_{\text{r}}^2}{2^t} + \frac{q_{\text{send}} + q_{\text{exe}}}{2^t} + (q_{\text{send}} + q_{\text{exe}}^2) \frac{1}{p},$$

where we use the Zipf model of Dodoneu in [8], where $|\mathcal{D}| = 187,901$ and $s = 0.753771$; $q_{\text{r}} = 2^3$; $\tau_{\text{e}}$ is the computation time for an exponentiation in $G$, 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-B.

**Theorem 2:** $G$, $D$, $n_0$ and $P$ 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 $q_{\text{send}}(\leq m_0 = 10)$ Send-queries and $q_{\text{exe}}$ Execution-queries, and making less than $q_{\text{r}}$ random oracle queries. Then we have

$$\text{Adv}_{\mathcal{A}}^{\text{P.D}}(\mathcal{A}) \leq \sum_{j=1}^{q_{\text{send}}} \frac{1}{|\mathcal{D}|} \frac{q_{\text{r}}}{n_0} + \frac{q_{\text{send}} + q_{\text{exe}}}{2^t} \frac{q_{\text{send}} + q_{\text{exe}}^2}{2p}$$

$$+ 5q_{\text{r}} \text{Adv}_{\mathcal{A}}^{\text{P,D}}(t + (q_{\text{send}} + q_{\text{exe}} + 1) \cdot \tau_{\text{e}}),$$

where $\tau_{\text{e}}$ is the computation time for an exponentiation in $G$ 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-B.

VII. **Performance evaluation**

In this section, we compare the performance and the fulfillment of the criteria among relevant schemes [30], [35], [36], [49], [57], [62]–[65] and our proposed scheme. The comparison results are depicted in Table V. Sixty-seven typical schemes are further evaluated in Appendix A.

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, timestamp values and output of secure one-way hash function are all recommended to be 128-bit long, while $y$ and $g$ are 1024-bit long. Let $T_{H}, T_{E}, T_{S}$, and $T_{C}$ denote the time complexity for hash, modular exponentiation, symmetric encryption and Chebyshev polynomial, respectively. Other lightweight operations like XOR and || are omitted.

<table>
<thead>
<tr>
<th>Table VI. Timings for cryptographic operations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experimental platform</strong></td>
</tr>
<tr>
<td>Philips HiPerSmart 136 MHz</td>
</tr>
<tr>
<td>Intel(R) T5820 2.00 GHz</td>
</tr>
<tr>
<td>Intel(R) i7-4790 3.60GHz</td>
</tr>
</tbody>
</table>

To have a more intuitive grasp on the computation overhead of our scheme, in Table VI we list the computation time for related cryptographic operations on different platforms. We use a Philips HiPerSmart card [66] to approximate user device, and the computation time of related operations is reported in [67]. 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 [67] make use of the standard cryptographic library MIRACL [68], which is a multi-precision integer and rational arithmetic C/C++ library.

As illustrated in Table V, our scheme provides all the twelve criteria while maintaining reasonable efficiency; all the other schemes fail to achieve at least one critical criterion due to the inherent security-usability conflict revealed in [25], because they only employ conventional cryptographic approaches.

### VIII. CONCLUSION

In this paper, we have taken a 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” [38], 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 [27]. 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.

### REFERENCES


Ding Wang received his B.S. Degree in Information Security from Nankai University in 2008. Currently, he is pursuing his Ph.D. degree at Peking University. He received two outstanding reviewer awards and authored an “ESI highly cited paper”. His research interests mainly focus on two-factor authentication and password analytics.

Ping Wang received his B.S. degree from University of Electronic Science and Technology of China in 1983, and Ph.D. degree from University of Massachusetts, USA, in 1996. Now he is a Professor at Peking University. He served as TPC co-chairs of several security conferences. His research interests include system security and distributed computing.