2009 Third International Conference on Emerging Security Information, Systems and Technologies

# Analysis of a password strengthening technique and its practical use

Bogdan Groza
Department of Automatics and Applied Informatics
Politehnica University of Timisoara, Faculty of Automatics and Computers
Timisoara, Romania
bogdan.groza@aut.upt.ro

*Abstract*— **Besides commonly used password strengthening techniques such as salting or repeated applications of a one-way function on the password, we account a less common procedure: the truncation of the output from a one-way function on the password. This technique is used in a Norwegian ATM and a similar method is part of an authentication protocol from Anderson and Lomas which makes use of collision-full hash functions. We depict a probabilistic bound on the probability of guessing the password in the Anderson-Lomas protocol and we propose some improvements on the protocol. Further, the improved protocol proves to be a good solution for a password based authentication between two devices that authenticate in the absence of a previously known secret or of a trusted third party. The protocol proves to have all the desired properties for this scenario.**

***Keywords-authentication; password; protocol;***

## I. INTRODUCTION

Although passwords offer the weakest level of security, they are still the most used authentication factor. This is because they can be easily memorized by humans and further used for authentications without requiring the possession of additional devices such as smart-cards, mobile phones etc. that can be used to compute one-time password or at least store randomly generated keys.

The main disadvantage of passwords is their lack of entropy which makes them vulnerable against exhaustive searches over the password space. Thus, an adversary can built a dictionary of passwords and search in it for the correct one if the protocol reveals all information necessary for the verification. Even more, in some situations, when randomness is absent in hiding the password, the adversary can save some of his computational time and use pre-computed dictionaries. These attacks are more severe, as a search over a pre-computed dictionary can be done in no-time by using binary search.

In order to overcome dictionary attacks, the first solution was introduced by Bellovin and Merrit [5] and since then several modern solutions with provable security appeared [4], [6]. However, in practice only elementary improvements for strengthening passwords are used, from which the most common are salting and repeated applications of a one-way function. These techniques are known as folklore and used for a long time; a research account of some password strengthening techniques is available in [1].

Besides salting and repeated applications of a one-way function, in this paper we account an interesting mechanism that can be used to increase password security: the truncation of the output from a one-way function on the password. This technique can be used to eliminate some attacks as an adversary cannot decide which is the correct password based on a single truncated output. The technique was initially proposed by Anderson and Lomas in a key-exchange protocol [2]. Further, a similar procedure was used in practice in a Norwegian ATM. In this paper we make an analysis of the probability for an adversary to find the password in this case and further we propose some modifications that can improve the protocol security. The step-by-step verification procedure that we introduce decreases the number of disclosed bits from a protocol run. Although the proposed improvement requires more communication sessions, it is more efficient with respect to the security level which matters most. We also propose an interesting scenario in which the improved protocol can be used: an authentication between two devices, in the absence of a previously known secret or of a trusted third party, by the use of a very low entropy password.

The paper is organized as follows. Section 2 discusses along the lines of password strengthening techniques: salting, multiple applications of a one-way function and truncation. Section 3 makes an analysis of guessing in the presence of truncation of the output from a one-way function on the password that is used in the Anderson-Lomas and the Norwegian ATM protocol. Further, section 3 proposes some improvements on this and introduces a practical scenario in which the protocol can be used. Section 4 holds the conclusions of our paper.

## II. WEAKNESSES AND IMPROVEMENTS OF SOME PASSWORD BASED PROTOCOLS FROM PRACTICE

### A. Compensating low-entropy with salting and repeated applications of a one-way function

Windows provides a good practical example of how wrong it is to forget about salting passwords. All Windows OS versions prior to Vista store user passwords as the *lmhash*, which consist in splitting the password of at most 14 chars in two values that are used to encrypt with DES the constant "KGS!@#$%", i.e. $DES_{pw1}("KGS!@\#\$\%") \| DES_{pw2}("KGS!@\#\$\%")$. In this case, besides dictionary

978-0-7695-3668-2/09 $25.00 © 2009 IEEE
DOI 10.1109/SECURWARE.2009.52

guesses, in which an adversary compares the password to entries in the dictionary, pre-computed dictionary attacks are feasible as the encryption is deterministic. These pre-computed dictionary attacks can be done even in a more specialized form known as rainbow tables, being efficient when cracking large amounts of passwords. Rainbow tables are distributed freely over the Internet and can be computed by using the advantage of grid systems or distributed computation communities. Free rainbow tables are available on the web to crack large amounts of Windows passwords.

To avoid pre-computed dictionary attacks, in UNIX OS salt is added to the passwords. Even more, passwords are hashed multiple times, making an adversary spent more time to find passwords by exhaustive search. It is also notable that in the traditional DES based crypt command from UNIX the encryption algorithm was also modified by the salt in order to block the potential use of standard cryptographic hardware.

In this context, it is also useful to mention the technique from Abadi et al. [1] that consists in concatenating a value to the password before hashing and delete this value afterwards. This means that the password is hashed as $H(pw,r)$ and if $r$ is unknown the system must verify all values of $r$ before accepting a password. Now, if $r$ has an appropriate size (for example 20 bits) the system will not need to much time to confirm the correct password but the adversary time needed to test many wrong password will increase proportionally. Finally, this has the same effect as the repeated application of a one-way function on the password.

### *B. Hiding passwords by truncating one-way functions outputs: the Norwegian ATM and Anderson-Lomas protocols*

There is a more subtle way of hiding passwords than salting or repeated applications of a one-way function. The main idea is that the password can be hidden by truncating the output from the one way function computed on the password.

An interesting example is from a Norwegian ATM authentication system, which was shown to be flawed after a trial from an honest user who lost a credit card from which money were subtracted [8]. Each credit card stores on it a 16 bit verification value that is the result of a 16 bit truncation from a DES encryption of the PIN with a key *K* known only by the bank and the ATM's, i.e. each card stores $\lfloor DES_K(PIN) \rfloor_{16}$. We simplified this here because the DES is not computed directly on the PIN and other information such as the account number is also encrypted, but since the password is the one on which we are interested, other details are not relevant. Now, if an adversary has a stolen card and wants to find the PIN, the main problem is that even if it makes an exhaustive search against the $2^{56}$ possible DES keys (which was feasible at that time and nowadays DES can be cracked in less than a week), since the value from the card is truncated to 16 bits it will get roughly $2^{40}$ correct DES keys which can further validate any PIN code. However, these false DES keys can be discarded if the adversary gets more honest credit cards from the same bank or changes its PIN code. This is because each new PIN provides him a new 16 bit value and by testing DES keys each such value reduces the number of correct DES keys by a factor of $2^{16}$. Thus, finding the correct DES key can be done after roughly 4 honest cards issued to the adversary and the PIN from the stolen card can be further found by testing against the 16 bit value from this card.

The same truncation is used in a protocol proposed by Lomas and Anderson where the collision-full hash functions proposed by Gong [7] are used to authenticate a key exchanged with Diffie-Hellman.

**Diffie-Hellman-Merkle Key-Exchange**

$$A \to B:\ \gamma^{\alpha}$$
$$B \to A:\ \gamma^{\beta}$$

**Anderson Lomas**

$$A \to B:\ H\left(H\left(pw,\gamma^{\alpha\beta}\right)\bmod 2^m,\gamma^{\alpha\beta}\right)$$
$$B \to A:\ H\left(H\left(H(pw),\gamma^{\alpha\beta}\right)\bmod 2^m,\gamma^{\alpha\beta}\right)$$

Here $\gamma^{\alpha}$, $\gamma^{\beta}$, $\gamma^{\alpha\beta}$ are the usual Diffie-Hellman parameters and $f_k(x)=H\left(H(k,x)\bmod 2^m,x\right)$ is the collision-full hash function. The collision-full hash has the property that for a target image $f_k(x)$ if one knows $x$ it is easy to find collisions in the first variable, i.e. $f_k(x)=f_{k'}(x),k\neq k'$, but if one knows $k$ it is infeasible to find collisions in the second variable, i.e. $f_k(x)=f_k(x'),x\neq x'$. As for the notation $H(x,y)$ in the original paper from Anderson and Lomas [2] the hash function is used over the concatenation of the two variables while later papers from different authors also interpret this as a hash function keyed on the first variable. Finally, in this context any of the interpretations can be used.

Thus, it is acknowledged in [2] that if $Adv$ plays as man-in-the-middle and impersonates $B$ he knows $\gamma^{\alpha\beta}$ but further he cannot find *pw* since there remain $2^{k-m}$ correct passwords after $Adv$ gets the response from $A$. It is easy to see the similarities between the Norwegian ATM and the Anderson-Lomas protocol, in what follows we are interested on computing some bounds on the probability of finding the password in these cases.

## III. Analysis and Improvements on the Anderson-Lomas Protocol

### *A. Probabilistic bounds on guessing the password*

Let us define $Adv_{guess}$ as the event that the adversary successfully guesses the password. Ideally speaking, this

probability must be negligible, but in practice it depends on the entropy of the passwords. If one considers passwords on $k$ bits and the target password is uniformly distributed among them we have $\Pr[Adv_{guess}] = 2^{-k}$.

First we want to show that using any truncated hash on the password and the Diffie-Hellman key is sufficient for the man-in-the-middle adversary in the Anderson-Lomas protocol while a collision-full hash is more than necessary (in particular we can renounce to the hardness of finding a collision in the second argument). Assume that $H$ is a collision-free hash function. For functions $f(pw,x) = H\left(H(pw,x) \bmod 2^m, x\right)$ and $g(pw,x) = H\left(H(pw), x\right) \bmod 2^m$ it is easy to show that finding $pw$ from a set of inputs-outputs $L_f = \left\{(x_i, y_i) \mid y_i = f(pw, x_i)\right\}$ or from a similar set $L_g = \left\{(x_i, y_i) \mid y_i = g(pw, x_i)\right\}$, happens with the same probability if the sets have the same size, i.e. $|L_f| = |L_g|$. To prove this, assume by contradiction that it is harder for $Adv$ to correctly identify $pw$ from $L_f$ than from $L_g$ when $|L_f| = |L_g|$. This means that there are more collisions in $L_f$ than $L_g$. However, observe that given any pair $(x_i, y_i)$ from $L_f$ it holds $f(pw', x_i) = y_i, pw' \neq pw$ if and only if $H(pw, x_i) \bmod 2^m = H(pw', x_i) \bmod 2^m$. This is easy to prove since if we assume that $H(pw, x_i) \bmod 2^m \neq H(pw', x_i) \bmod 2^m$ it follows that the hash function has collisions which is a contradiction. But this means that the collisions in $L_f$ occur if and only if the same collisions occur in $L_g$. Therefore $pw$ can be correctly recovered from $L_f$ if and only if it can be correctly recovered from $L_g$.

This means that in the Anderson-Lomas protocol the man-in-the-middle attack works with the same strength even if we do not use the collision-full hash function and we simply use a hash function with a truncated output as follows:

**Truncated Hashing**

$$A \to B:\ H\left(pw, \gamma^{\alpha\beta}\right) \bmod 2^m$$
$$B \to A:\ H\left(H(pw), \gamma^{\alpha\beta}\right) \bmod 2^m$$

Now we want to give a probabilistic measure for the event $Adv_{guess}$. Obviously, if there are $n_{cand}$ candidate passwords and the correct password is randomly distributed between them, it holds:

$$\Pr[Adv_{guess}] = \frac{1}{n_{cand}} \tag{1}$$

Now we want to establish how $\Pr[Adv_{guess}]$ varies with protocol runs when $Adv$ impersonates the honest users and knows the Diffie-Hellman key. This will be done by estimating the number of candidate passwords that remain after some protocol runs. We will consider the general case of a pseudorandom function of two arguments $f: \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^m$ with $k > m$ (this includes the hash function from Anderson-Lomas, as well as the DES encryption from the Norwegian ATM). Further, we assume that the correct password is $pw \in \{0..1\}^k$ and the adversary knows pairs $r_i, f(pw, r_i), i = \overline{1,q}$ (where $r_i$ is some random value) and can compute $f(p, r_i), i = \overline{1,q}, \forall p \in \{0,1\}^k$. We want to establish how $\Pr[Adv_{guess}]$ varies after testing candidate passwords against the $q$ outputs for the correct password. Let $p_{coll}$ be the event that the output of $f$ on some candidate password and some additional input $r_i$ collide with the target, i.e. $f(pw, r_i) = f(p, r_i)$. Since $f$ is pseudorandom we have $\Pr[p_{coll}] = \frac{1}{2^m}$ and after testing against all the pairs $r_i, f(pw, r_i), i = \overline{1,q}$ the probability for a password to collide on all $q$ targets is $\Pr[p_{q-coll}] = \frac{1}{2^{mq}}$. Therefore in average the number of candidate passwords will be:

$$n_{cand} = 1 + \left(2^k - 1\right) \cdot \Pr[p_{q-coll}] = 1 + \frac{2^k - 1}{2^{mq}} \tag{2}$$

Here 1 stands for the fact that there will be at least one collision, and this is with the correct password. This gives the following probability for the adversary to guess the correct password:

$$\Pr[Adv_{guess}] = \frac{2^{mq}}{2^{mq} + 2^k - 1} \tag{3}$$

*B. Further guessing informations*

The calculus for relation (3) has taken into account only the values provided by $A$ to $Adv$. However if $Adv$ plays as man-in-the-middle besides the correct values provided by $A$, $Adv$ may further try to give a correct response to $B$. If he gives a correct response then $Adv$ learns two more input-output pairs for the function computed on the secret. However, $Adv$ can give the correct answer only with

probability $2^{-m}$ which should be negligible in practice since otherwise *Adv* can successfully impersonate $A$. On the other hand, with probability $1-2^{-m}$, *Adv* gives the wrong answer to $B$ and in the case that $B$ will not reply he learns that the value does not match the correct password. Now, if the password space is $2^x$ he can further exclude $2^{x-m}$ passwords, however this is a reduction of the password space with a factor of $1-2^{-m}$ which is again negligible compared to the reduction by a factor of $2^{-m}$ that results from the response provided by $A$. Therefore, it seems to be little point for *Adv* to play the role of $A$ to $B$ as it is very likely that $B$ detects its presence while he learns almost nothing about the password. This means that it is sufficient to approximate the number of candidate passwords based only on the responses from $A$. Nevertheless this shows that *Adv* does not have good chances in defeating the protocol if he is the initiator of the protocol.

Still, in the best case it is desirable for a protocol to let *Adv* test only one password in each protocol run (i.e., in each on-line attack). However, in this protocol, after each run as the initiator, *Adv* can exclude about $2^{x-m}$ passwords as previously shown. Therefore we can consider that the protocol is not optimal against on-line attacks.

### C. On the optimal choice of m

In the original proposal from Anderson and Lomas [2] there is a discussion on the optimal choice of $m$ and the recommend value is $m = k/2$. Indeed, we may consider that after the first response from $A$ to *Adv* there are two ways of attacking the protocol: first by guessing the password from the response of $A$ (which happens with probability $2^{m-k}$) and second by giving the correct response to $B$ which happens with probability $2^{-m}$. If one wants to minimize the probability of both the attacks the optimum choice will be indeed $m = k/2$.

Still, choosing such a large $m$ will lead very fast to the correct value of the password, forcing to stop the protocol quickly in the case of an attack or otherwise the password will be guessed. It may be useful to further improve on this by dynamically adjusting $m$. For example, one may choose to reduce $m$ to half after each wrong response from the other party (which indicates the potential presence of the adversary). This will force to stop the protocol after about $\log_2 k$ unsuccessful runs as $m$ cannot be further reduced. At this point *Adv* may have successfully find the password with probability $2^{-1}$. The only advantage is that the probability of guessing increases more smoothly to $2^{-1}$. Note however that now the probability that *Adv* gives a correct response to $A$ also gets double each time $m$ is reduced by half. So more likely such an improvement will not help much. $\Pr[p_{q-coll}]$ for this case will be:

$$\Pr[p_{q-coll}] = \frac{1}{2^{m+\frac{m}{2}+\frac{m}{4}+\ldots+\frac{m}{2^{q-1}}}} = 2^{m\cdot\frac{1-2^q}{2^{q-1}}} \quad (4)$$

Further, this relation can be used in (2) to compute $\Pr[Adv_{guess}]$. In figure 1 at the end of this section there are comparative plots for the variation of $\Pr[Adv_{guess}]$ with protocol runs and the case of the dynamic adjustment for $m$ is outlined as well.

### D. A further improvement on the Anderson-Lomas protocol

The main problem in the Andreson-Lomas is that when *Adv* plays the role of $B$, $A$ discloses the value of $H(MAC_{pw}(\gamma^{\alpha\beta}) \bmod 2^m, \gamma^{\alpha\beta})$ which lets *Adv* reduce the password space by a factor of $2^m$. We now improve on this by proposing a step-by-step verification procedure for which we show that it reduces the probability that *Adv* finds the correct password. $A$ and $B$ compute the regular values (or even the same value) and disclose only a fixed number of bits in each round. Ideally for security, only 1 bit should be disclosed; however, this increases the number of sessions. For example, $A$ and $B$ can compute $v = v_1 v_2 \ldots v_{r\cdot s} = MAC_{pw}(\gamma^{\alpha\beta}) \bmod 2^m$ and $w = w_1 w_2 \ldots w_{r\cdot s} = MAC_{H(pw)}(\gamma^{\alpha\beta}) \bmod 2^m$ then proceed to the following step-by-step verification:

**Step-by-step Verification**

$$\begin{aligned} &Session\ \ i = 1..r \\ &A \rightarrow B:\ v_{(i-1)\cdot s+1} \ldots v_{i\cdot s} \\ &B \rightarrow A:\ w_{(i-1)\cdot s} \ldots w_{i\cdot s} \end{aligned}$$

Now, if *Adv* plays the role of $B$ then in session 1 he receives $s$ bits from $A$ and afterwards he will get to session 2 only if he responds correctly to $A$ in session 1. Assuming that the password is unknown to *Adv* this happens with probability $\Pr[Adv_{corr}] = 2^{-s}$. We can compute the probability that Adv stops in some session $i = 1..k$, i.e. the event that *Adv* is correct up to session $i$ denoted $Adv_{i-corr}$, as:

$$\Pr[Adv_{i-corr}] = \left(\frac{1}{2^s}\right)^{i-1} \cdot \left(1 - \frac{1}{2^s}\right) \quad (5)$$

We now want to compute the average number of bits which *Adv* gets after one protocol run. In the first session if *Adv* plays the role of $B$ he gets $s$ bits from $A$, afterwards he will get another $2 \cdot s$ for any correct answer. Therefore the average number of bits *Adv* gets from $A$ is:

$$av_A = s + \sum_{i=1}^{r-1} 2 \cdot i \cdot s \cdot \left(\frac{1}{2^s}\right)^i \cdot \left(1 - \frac{1}{2^s}\right)$$
$$= s + 2 \cdot s \cdot \left(1 - \frac{1}{2^s}\right) \sum_{i=1}^{r-1} i \cdot \left(\frac{1}{2^s}\right)^i \quad (6)$$

By elementary computations we can compute $\sum_{i=1}^{k} i \cdot x^i = \frac{k \cdot x^{k+2} - (k+1) \cdot x^{k+1} + x}{(x-1)^2}$ and by replacing in the previous relation this gives:

$$av_A = s + 2 \cdot s \cdot \frac{2^{-r \cdot s} \cdot \left(r \cdot 2^s - r + 1\right) - 1}{1 - 2^s} \quad (7)$$

However we should not forget what happens if *Adv* plays the role of $A$ towards $B$. *Adv* can give $B$ a correct response with probability $2^{-s}$, but different to the case of the Anderson-Lomas protocol, this may not be negligible anymore. Therefore, the average number of bits received from $B$ is:

$$av_B = \sum_{i=1}^{r} 2 \cdot i \cdot s \cdot \left(\frac{1}{2^s}\right)^i \cdot \left(1 - \frac{1}{2^s}\right) \quad (8)$$

By adding $av_A$ and $av_B$, in one protocol run if *Adv* plays the man-in-the-middle he will get the following average number of bits:

$$av \approx s + 4 \cdot s \cdot \frac{2^{-r \cdot s} \cdot \left(r \cdot 2^s - r + 1\right) - 1}{1 - 2^s} \quad (9)$$

In (9) for simplicity we have omitted the last term from the sum in $av_B$ which is negligible. Further, the average number of bits can be replaced in relation (3) where the probability was computed for the case when $m$ bits were disclosed, giving:

$$\Pr[Adv_{guess}] = \frac{2^{av \cdot q}}{2^{av \cdot q} + 2^k - 1} \quad (10)$$

Basically the effect of the step-by-step verification is the same as if in the original protocol the size of $m$ will be reduced to the average value from (9) and the protocol is run $m / av$ times. However the advantage of the step-by-step verification is that one does not need to repeat the hash computations required by each protocol run. Figure 1 shows the plots done in Mathematica for the variation of probability for guessing based on the number of protocol runs on all the cases of the protocol that were analyzed. We underline that by using the step-by-step verification procedure there is no need to truncate the hash since the probability for *Adv* to get all bits from the image of the hash is negligible anyway.

### *E. A potential application*

We outline a particular scenario that can benefit from such a password strengthening technique: an interaction between two mobile devices handled by users who choose weak passwords. One may consider that the two devices are for example mobile phones and two users want to share content between them. As there is no previously shared key between the devices and a trusted third party does not exist, the only way is for the users to verbally agree on a password that is set on both devices. However, this is not enough, since if the password is weak and the protocol exposes information that makes the password guessable, or even more gives the opportunity to search it via a pre-computed dictionary, there will be no security for the communication.

Using the improved protocol is efficient and has at least two merits. First is that pre-computed dictionary attacks are not feasible as the Diffie-Hellman key exchange assures fresh random information each time. Second, and more important, a potential adversary playing as man-in-middle will be quickly detected since it will fail to give the correct response with high probability. Finally, if the adversary plays as man-in-the-middle, it can not recover the password and will get more passwords that correspond to the captured values. Further, the two participants are warned that they are under attack and they will choose a new password.

For example, one can consider that the password has 20 bits (which is not much) and choose $m = 10$ in the original Anderson-Lomas protocol. This will lead to a probability of $2^{-10}$ for *Adv* to bypass the authentication and with probability $1 - 2^{-10}$ his tentative will be detected. More, if the proposed step-by-step verification is used one can set $m = 20$, which is the same size as the password and means that there is no truncation, but if $r = 10$, $s = 1$ then *Adv* gets only 5 bits in average and the probability for giving a correct response is only $2^{-20}$, i.e., less than one in a million.

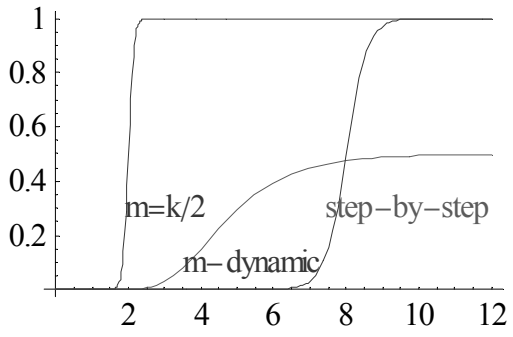


Figure 1. Probability that Adv guesses the password (x-axis depicts the number of man-in-the-middle attacks on the protocol and y-axis the guessing probability).

## IV. CONCLUSIONS

Users tend to choose weak passwords; still there are a number of improvements that can be used in practice to increase their strengths. Since passwords continue to be used, password strengthening techniques are playing an important role in security. Some techniques are quite ingenious as the truncation from the Norwegian ATM or Anderson-Lomas protocols. We showed that further improvements can be done on the Anderson-Lomas protocol and we made a precise analysis on the probability of guessing the password. Further we proposed the use of this protocol as a solution for a practical scenario where two users choose weak passwords to authenticate two mobile devices, the practical implementation of the protocol for a real-world scenario remains as potential future work for us.

## ACKNOWLEDGMENT

This work was partially supported by national university research council CNCSIS grant PNCDI PN II - 940/2009.

## REFERENCES

[1] M. Abadi, T. Lomas, R. Needham, Strengthening passwords, SRC Technical Note 1997 – 033, Digital Equipment Corporation, Systems Research Center, 1997.

[2] R. J. Anderson and T. M. A. Lomas. Fortifying key negotiation schemes with poorly chosen passwords. Electronics Letters, July 1994.

[3] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On password-based authenticated key exchange using collisionful hash functions. In Proc. First Australasian Conf. on Information Security and Privacy, volume 1172 of LNCS, Springer, 1996.

[4] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In Proc. Int'l. Conf. on the Theory and Application of Cryptographic Techniques (EuroCrypt), volume 1807 of LNCS, Springer, 2000.

[5] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In Proc. IEEE Symposium on Security and Privacy, 1992.

[6] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In Proc. 21st Ann. Int. Cryptology Conf., volume 2139 of LNCS, Springer, 2001.

[7] L. Gong. Collisionful keyed hash functions with selectable collisions. Information Processing Letters, 1995.

[8] K. J. Hole, V. Moen, A. N. Klingsheim, and K. M. Tande. Lessons from the Norwegian ATM system. IEEE Security and Privacy, 2007.