Public-Key Cryptography and Password Protocols: The Multi-User Case

Maurizio Kliban Boyarsky*

Abstract

The problem of password authentication over an insecure network when the user holds only a human-memorizable password has received much attention in the literature. The first rigorous treatment was provided by Halevi and Krawczyk, who studied off-line password guessing attacks in the scenario in which the authentication server possesses a pair of private and public keys. In this work we:

- Show the inadequacy of both the HK formalization *and* protocol in the case where there is more than a single user: using a simple and realistic attack, we prove failure of the HK solution in the two-user case.
- Propose a new definition of security for the multi-user case, expressed in terms of transcripts of the entire system, rather than individual protocol executions.
- Suggest several ways of achieving this security against both static and dynamic adversaries.

In a recent revision of their paper, Halevi and Krawczyk again attempted to handle the multi-user case. We expose a weakness in their revised definition.

1 Introduction

The problem of password authentication over an insecure network when the user holds only a human-memorizable password has received much attention in the literature. This is a realistic scenario for remote login, and it is extremely important both to correctly define the security in such settings and to prove the security of any proposed implementation. Consider the asymmetric scenario in which the authentication server possesses a pair of private and public keys while the user holds only a weak, memorizable password. A first step towards a rigorous treatment of this scenario was provided by Halevi and Krawczyk [17], who studied the case where there is a single user in the system. We continue this line of investigation.

A natural solution in the setting under consideration is an encrypted challenge-response protocol. Roughly speaking, the server generates a public key, private key pair. The users are all informed of the public key. When a user u attempts to log in, the server sends in the clear a challenge r to u, and u must reply with an encryption, under the server's public key, of u, r, and u's secret password, denoted $spwd_u$. The critical question in making such an approach work is the type of security that must be enjoyed by the cryptosystem. To see this, suppose the cryptosystem is *malleable*[1], and consider the case in which u, in response to a challenge r, responds with an encryption $c \in_R E(spwd_u, u, r)$. Then v, impersonating u and receiving a challenge r', may be able to transform c to $c' \in E(spwd_u, u, r')$. This is *trivial* if encryption is bit-by-bit, so semantic security is clearly insufficient (because there exist semantically secure bit-by-bit encryption schemes, *e.g.* [13]). As this example shows, some non-malleability is essential to implementing the encrypted challenge-response approach.

Halevi and Krawczyk first present definitions of a *forging attack* against a password authentication scheme in this scenario, and of what it means to break a scheme. They next introduce the notion of a public-key cryptosystem that is semantically secure against a weak type of chosen-ciphertext attack they call a *"one-ciphertext verification"* attack, and present the encrypted challenge-response password authentication scheme. Finally, they prove that if the authentication scheme is implemented using a public-key cryptosystem that is is semantically secure against one-ciphertext verification attacks then it is secure against their notion of a forging attack. We note that any cryptosystem that is non-malleable against a chosen-plaintext attack is semantically secure against one-ciphertext verification attacks (Theorem 7.1).

Halevi and Krawczyk's definition of security concentrates on the single-user (and single server) case. One may suspect that, as a general principal, a protocol which is good when there is one user is also good when there are

*Mailing address: Fine Hall, Dept. of Mathematics, Princeton University, Princeton, NJ 08544, USA. E-mail: mkboyarsky@yahoo.com. Supported by ESPRIT Grant RAND2.

[1] Non-malleability, an extension of semantic security defined in [11], says, roughly, that for any polynomial-time computable relation R, seeing $E(x)$ "does not help" in finding an *encryption* $E(y)$ such that $R(x, y)$ holds.

several. This turns out to be false: we show that there are protocols secure in the single-user case but insecure in the multi-user case (Section 3). In particular, there exists a public-key cryptosystem that is semantically secure against one-ciphertext verification attacks (the notion proposed in [17]), but under which the HK challenge-response password authentication system is easily broken in the two-user case (we exhibit an explicit break). The control needed by the adversary for our attack is minimal and realistic: it does not require an active intruder-in-the-middle; the adversary simply has to eavesdrop to a single authentication session of a good user and control one other user.

On a more positive note, we provide a definition of security for the multiple-user case (Section 4), and outline several protocols that satisfy it, including the proposal of Halevi and Krawczyk when the requirements for the encryption scheme are strengthened. We address both static and dynamic adversaries in the multi-user case. These results appear in Section 5.

In Section 6 we map the complexity of the cryptographic primitives that must be used in order to achieve secure password authentication in various settings. We give neccesary and sufficent conditions for both the case where the user knows (or can verify) the public key of the server, and for the case where no such key is known.

Independently of this work, Halevi and Krawczyk considered the multi-user case as well [18], revising their definition of security and changing the requirements for the public-key encryption function. In Section 8 we discuss some shortcomings of their revision.

1.1 Related Work

As noted above, the problem of authentication with human-memorizable passwords has an extensive history (see, for example, [6, 15, 16, 17, 28]). The goal is to (safely) use a not-so-long and not-so-random password to allow authentication over a hostile network, where a particularly problematic issue is off-line guessing of passwords. To the best of our knowledge, Halevi and Krawczyk were the first to consider this problem in a rigorous manner, without making any *idealized* assumptions about encryption functions.

In a related line of work, Bellare and Rogaway considered authentication in several settings [3, 2, 4]. We apply some of their ideas in order to define security in the multi-user case. Resilience against adaptive adversaries in general multi-party protocols has been studied in several places (see, *e.g.*, [7, 8]). We use a notion of *non-committing encryptions*, developed for these purposes, in order to handle the case of adaptive corruption of users. It can also be used to provide *forward secrecy*.

2 Precise Description of the Halevi-Krawczyk Results

In this Section we review the results in [17]. Section 2.1 reviews the HK definition of security for a password authentication scheme (both attack and break); Section 2.2 describes their notion of semantic security for a cryptosystem under one-ciphertext verification attacks; Section 2.3 describes a natural "generic" encrypted challenge-response protocol for password authentication. In Theorem 1 of [17] it is shown that when the generic protocol is implemented using a cryptosystem semantically secure against one-ciphertext verification attacks, then the protocol satisfies the HK definition of security for password authentication.

2.1 Security of Password Authentication Schemes

Halevi and Krawczyk do not provide protocols for system setup: it is simply assumed that the server's public key is established and a hash, under a collision-intractable hash function, is known to the user (there is a nice discussion of this in [17]); similarly, the user's password is established and known to both the server and the user.

Halevi and Krawczyk consider the case in which the attacker, referred to as a *forger*, can be quite active. We quote from [17]:

Definition 2.1 *(HK forger's attack): "The attacker ... is allowed to watch regular runs of the protocol between the* user U *and the* server S*, and can also actively communicate with the user and the server in replay, impersonation and man-in-the middle attacks. The forger can initiate multiple authentication sessions with S as if they were requested by U. It can intercept challenges sent from S to U and change them to any value of its choice, and then see the response from U to that (possibly modified) challenge and change their values and then send the (possibly modified) response to S. Finally, F also gets to see whether S accepts the authentication or not."*

Definition 2.2 *(HK break of a password scheme): Halevi and Krawczyk say that the forger* breaks *the authentication protocol if "S accepts a response which was sent to it by F but was not freshly generated by U."*

[17] assumes a relatively small password *dictionary*, $\mathcal{D}$. Since $\mathcal{D}$ is assumed not to be prohibitively large, the system can be broken by exhaustive search of the dictionary. The HK definition of security says, intuitively, that the forger should not be able to break the system with considerably less effort than that needed for such a brute force approach:

Definition 2.3 *(HK definition of security for a password verification scheme; [17] Definition 1): Let ε be any positive real number. We say that a one-way password authentication protocol enjoys basic security up to ε, if no feasible attacker F can break the protocol with probability higher than $\frac{v}{|\mathcal{D}|} + \varepsilon$, using only v impersonation attempts with the server.*

Halevi and Krawczyk explain that "feasible" should be construed to potentially permit exhaustive search over the password dictionary (if it is sufficiently small), but not to permit breaking of the cryptographic primitives used in the protocol (footnote 4, [17]).

Limitation: The starting point for our work is the observation that Halevi and Krawczyk's definition of an attack (Definition 2.1) does not permit F to interact with the server *except when posing as U*. In particular, if F is itself a valid user, the attack does not permit F to interact with the server while claiming to be itself. As we shall see, this is a crucial difference.

A more subtle point is that the Halevi-Krawczyk definition of security (Definition 2.3), when generalized to

the multi-user case in the natural way, ignores situations where, for example, two user's passwords may be related. Indeed, it is precisely this point that is overlooked in the revised paper [18]. We say more about this in Section 8. Our definition (Section 4.2) handles this.

2.2 One-Ciphertext Verification Attacks

A cryptosystem $\mathcal{E}$ is specified by a triple of procedures (G, E, D). G is the key-generation algorithm, E is the encryption algorithm, and D is the decryption algorithm. Halevi and Krawczyk define a weak type of chosen ciphertext attack that they call *one-ciphertext verification* attacks. We paraphrase [17] in describing this attack. We have named certain parts of the adversary B_1, B_2, and B_3 for later use. We let $A = (B_1, B_2, B_3)$, with complete communication between the different B_i.

1. The key-generation algorithm is run (with security parameter k), to generate a secret/public key pair, (S, P).
2. Given P, the adversary B_1 generates a message s.
3. Let $r \in_R \{0,1\}^{|s|}$. $z \in_R \{s, r\}$ is chosen and B_2 is given $c \in_R E_P(z)$. B_2 generates a query (x', c'), where $c' \neq c$.
4. B_3 is told whether or not $x' = D_S(c')$ and guesses whether or not $c \in E(s)$.

The following is a slight modification of [17] Definition 2. The modification has no impact on our results.

Definition 2.4 *An encryption scheme (G, E, D) is said to* resist one-ciphertext verification attacks *if for any probabilistic poly-time bounded adversary $A = (B_1, B_2, B_3)$:*

$$\begin{aligned} & |Pr[A \text{ guesses “encryption of x ”} \mid c \in E_P(x)] \\ - \;& Pr[A \text{ guesses “encryption of x ”} \mid c \in E_P(r)]| \\ \leq \;& \nu(k) \end{aligned}$$

where the function ν grows more slowly than the inverse of any polynomial, k is the security parameter given to the public-key cryptosystem generator in generating the S/P pair, and the probabilities are taken over the execution of G, the coins of A, and the randomness used in Step 3 of the attack.

2.3 The Challenge-Response Password Authentication Protocol

The *Generic Encrypted Challenge-Response Protocol* of [17] is described next. It is assumed that, during set-up, the user learns a so-called “public password” for the server, which is a collision-intractable hash of the server's public key.

1. $S \longrightarrow u$: Pick a nonce r and send $\langle r, P\rangle$, where P is the public key of the server.
2. $u \longrightarrow S$: Check that P hashes to the hash value learned during set-up. If not, then abort. Else send $c \in_R E_P(f(spwd; r, u, S))$, where *spwd* is the user u's secret password, S is the name of the server, and f is a function of slightly special structure.

We do not go into the details of the structure of f here, except to note that [17] give the *concatenation* function as an example. For the rest of the paper we will assume that f is indeed the concatenation function.

The following theorem is proved in [17] (the actual text differs insignificantly from the version stated here). As we will prove (Section 3.2), the theorem holds only in the *single user* case, in which the adversary satisfies Definition 2.1.

Theorem 2.1 *([17] Theorem 1): Let $\mathcal{E}$ be an encryption scheme that resists one-ciphertext verification attacks and let f be one-to-one on its components (e.g., f may be the identity function). Then the encrypted challenge-response protocol using $\mathcal{E}$ and f enjoys basic security up to $M \cdot \nu(k)$, where M is at most quadratic in the number of messages that are sent by the attacker during active impersonation attacks against the protocol, and k is the security parameter.*

Let $\mathcal{G}$ denote the generic encrypted challenge-response protocol. When $\mathcal{G}$ is implemented with a public-key cryptosystem $\mathcal{S}$ we write $\mathcal{G}(\mathcal{S})$. Thus, Theorem 2.1 says roughly that (for appropriate choice of f) $\mathcal{G}(\mathcal{E})$ satisfies Definition 2.3.

Remark 2.2 *If a forger acts as a clear channel between S and u in the HK protocol, then under Definition 2.2, no break has occurred. Suppose the public-key cryptosystem used by the server is malleable only in the following sense: given an encryption $c \in_R E_P(f(spwd; r, u, S))$ it is easy to find $c' \neq c \in_R E_P(f(spwd; r, u, S))$ (that is, it is easy, given only the ciphertext, to find a* different *encryption of the same plaintext). Then if u', being given c, sends c' to the server, this* is *considered a break. Our definition, presented next, avoids this inconsistency.*

3 Multi-User Insecurity of the HK Protocol

In this section we present a public-key cryptosystem, $\mathcal{S}$, that we prove semantically secure under a one-ciphertext attack. Thus, this scheme satisfies Definition 2 of [17] (Definition 2.4 above). The scheme can be viewed as a watered down version of the [11] public-key cryptosystem secure against chosen ciphertext attacks in the post-processing mode. We then exhibit a simple break of $\mathcal{G}(\mathcal{S})$ (the implementation of the encrypted challenge-response password authentication protocol with cryptosystem $\mathcal{S}$).

3.1 The Scheme $\mathcal{S}$

We first describe the cryptosystem at a high level, then discuss the tools needed for the construction, and finally present the details.

High Level Description of $\mathcal{S}$ The public key consists of a collection of n pairs of keys $\langle e_i^0, e_i^1\rangle$, $i = 1, \ldots, n$, and a universal one-way hash function h, providing a mapping that defines a choice of a subset of the encryption keys.

The process of encrypting a message consists of the following steps:
(1) Choose an “identity” for the current encryption by generating a public signature verification key F; the corresponding signing key is kept private.

(2) Split the message into n strings whose bit-wise Xor equals the message.
(3) Encrypt the n strings using n encryption keys chosen from the set of keys according to the bits of $h(F)$, where F is the public signature verification key of Step 1.
(4) Sign the resulting encryptions using the private signing key chosen in the first step.

When a message is decrypted it must first be verified that the signature is proper. If not, then the output is null.

The Tools. We require a *single-bit* probabilistic public-key cryptosystem that is *semantically-secure against a chosen plaintext attack*. Let GP denote the key generator, e and d denote the public and private keys, respectively, and E and D denote, respectively, the encryption and decryption algorithms. The encryption of a string x will be bit-by-bit. The cryptosystem uses a *universal family of one-way hash functions* as defined in [26]. This is a family of functions H such that for any x and a randomly chosen $h \in_R H$, the problem of finding $y \neq x$ such that $h(y) = h(x)$ is intractable. Finally we need a one-time *signature scheme*, which consists of GS, the signature scheme generator that outputs F, the public key verification key, and P the private signing key (see e.g. [26]).

Detailed Description of $\mathcal{S}$:

Key generation: Run $GP(n)$, the probabilistic encryption scheme generator, $2n$ times. Denote the output by

$$(e_1^0, d_1^0), (e_1^1, d_1^1), (e_2^0, d_2^0), (e_2^1, d_2^1), \ldots (e_n^0, d_n^0), (e_n^1, d_n^1).$$

Generate $h \in_R H$. The public encryption key is $\langle h, e_1^0, e_1^1, e_2^0, e_2^1, \ldots, e_n^0, e_n^1 \rangle$. The corresponding private decryption key is $\langle d_1^0, d_1^1, d_2^0, d_2^1, \ldots d_n^0, d_n^1 \rangle$.

Encryption: To encrypt a k-bit message $m = b^1 \ldots b^k$:

1. Select random bits b_j^i for $1 \leq i \leq k$ and $1 \leq j \leq n$ such that $\oplus_{j=1}^n b_j^i = b^i$ for all $1 \leq i \leq k$.
2. Run $GS(n)$, the signature key generator. Let F be the public signature verification key and P be the private signature key.
3. Compute $h(F)$. Denote the output by $v_1 v_2 \ldots v_n$ where each $v_j \in \{0, 1\}$.
4. For each $1 \leq i \leq k$ and for $1 \leq j \leq n$ generate c_j^i, an encryption of b_j^i using $e_j^{v_j}$.
5. Create a signature s of the sequence $\{c_j^i\}_{\substack{1 \leq i \leq k \\ 1 \leq j \leq n}}$.

The ciphertext is $\langle F, s, c_1^1, \ldots c_1^k \ldots, c_n^1 \ldots c_n^k \rangle$.

Decryption: To decrypt $\langle F, s, c_1^1, \ldots c_1^k \ldots, c_n^1 \ldots c_n^k \rangle$:

1. Using public signature verification key F, verify that s is a signature of $\{c_j^i\}_{\substack{1 \leq i \leq k \\ 1 \leq j \leq n}}$.
2. Compute $h(F)$. Denote the output by $v_1 v_2 \ldots v_n$.
3. If the signature passes the verification, then for all $1 \leq i \leq k$ and all $1 \leq j \leq n$ retrieve b_j^i by decrypting using $d_j^{v_j}$. For $i = 1, \ldots, k$, let $b^i = \oplus_{j=1}^n b_j^i$ and let the decrypted message be the k-bit string $m = b^1 b^2 \ldots b^k$. Otherwise the output is null.

Theorem 3.1 *$\mathcal{S}$ is resistant to a one-ciphertext verification attack.*

The proof is omitted for lack of space.

3.2 Insecurity of $\mathcal{G}(\mathcal{S})$ in the Multi-User Case

Assume the adversary has control over a single compromised user u'. Let u be a non-compromised user. Given a ciphertext $\langle F, s, c_1^1, \ldots c_1^k \ldots, c_n^1 \ldots c_n^k \rangle$ of $(spwd_u, r, S, u)$, the adversary learns b_j^i for all $1 \leq i \leq k$ and $1 \leq j \leq n$.

This is done as follows for each i and j in its turn when u' is given a challenge and should respond:

- Generate a signature key F' such that the jth bit of $h(F')$ equals the jth bit of $h(F)$ - this is true for half the F''s, so it is easy to generate.
- Substitute c_j^i instead of the ciphertext $(c')_j^i$ that was suppose to be in the corresponding place in the reply by u'.
- Get through u' whether the response was accepted or not. Since $(b')_j^i$ (the bit encrypted by $(c')_j^i$) is known to u', it is possible to deduce the value of b_j^i.

The intrusion detection parameter can be re-set to zero as needed by having u' execute a successful login (as itself).

4 Our Definition of Security in the Multi-User Case

As always in defining security, we must specify the attack to be defended against, as well as what it means to break the system. For the rest of the paper, the term "compromised user" includes any party under the control of the adversary (such as the *forger* in the HK definitions), and is not restricted to users that do not have valid passwords.

4.1 Description of the Adversary in the Multi-User Case

Our notion of attack against a password authentication system is the HK definition generalized to handle the multi-user case (and, for the positive results, generalized to handle arbitrary password distributions). As in [17], we simply assume that the server's public key is established and a hash of this key under a collision-intractable hash function is known to the users. We assume that each user knows its own password, and (as in [17]), the passwords are known to the server. (Hence, it doesn't make sense to discuss the case of a corrupted server.) Our break of the HK protocol in the multi-user case is independent of whether the passwords are chosen by the users or by the server. Our positive results hold in either case as well; in particular they hold in the case in which the adversary has the additional power of choosing the passwords of the compromised users. Also as in [17], we assume a single server (our results generalize to multiple servers with access to the same passwords but with possibly different public keys). Unlike in [17], we assume multiple users.

For most of the paper we assume a *static* polynomial-time bounded (in the security parameter) adversary: the adversary chooses which users to compromise (make faulty) before execution of the protocol begins. The case of the *dynamic* adversary is extremely interesting. We discuss this case and offer a solution in Section 5.2. The adversary may compromise any number of users, and completely controls the behavior of the compromised users. The compromise may take place before the users have chosen their passwords, and the adversary may completely control the selection of passwords by the compromised users. However, as mentioned, we assume those compromised users having passwords are aware of their passwords.

The users in the system choose their passwords according to some joint distribution

$$\mathcal{D} = (D_1, D_2, \ldots D_{|U|})$$

where U is the universe of users. User u gets his password from distribution D_u. We assume nothing about the joint distribution; for example, the passwords of two users might be correlated or even identical.

The system may have some sort of *intrusion detection policy*, based on the number and frequency of failed attempts. For simplicity we assume that the system has an intrusion detection parameter I. This can be used in one of two ways: user u's account is frozen (u cannot log in without appeal to some restoration protocol) after either I consecutive failed attempts or I failed attempts since the last change of password. Our demonstration of the insecurity of the HK protocol in the multi-user environment (Section 3) is simplified if only consecutive failed attempts are counted, so we will make that assumption here. However, the security of our solutions for a multi-user environment (Section 5) can withstand any intrusion-detection policy and the policy need not be specified explicitly.

The adversary may listen to any number of successful login attempts (challenges and encrypted responses) by other users; the adversary can initiate any number of authentications, posing either as non-compromised processors or compromised processors; it can intercept and alter messages and generate spurious messages, in any direction between the server and any user, and it can learn if attempted authentications were successful. We also assume that the adversary has complete control over timing of events in the system.

4.2 Definition of a Break in the Multi-User Case

Let $\mathcal{A}$ be any adversary, for example, as described in Section 4.1. We consider long-lived runs of the password authentication protocol (the runs have length bounded in the security parameter). Each run consists of many invocations of the password authentication protocol, some of which may fail, together with other spurious messages that may be generated by the adversary and any replies to these messages. Thus, each long-lived run gives rise to a transcript.

Ideally, we would like to say that the system is secure if the adversary cannot make the server accept a session not initiated by said user, incorporating some notion of matching between the user's initiations and the server's acceptance, as was done by Bellare and Rogaway in several settings [3, 4]. Indeed, if the passwords were chosen uniformly from a very large space, such a definition is reasonable. However, since we assume the passwords are drawn from a distribution that is weak against repeated sampling or exhaustive search, some impersonation will occur in practice. Our definition, like the definition in [17], must reflect this and make sure that the protocol itself doesn't add to the insecurity.

Intuitively, we can say that the system is secure if the transcripts of the login session can be simulated without access to the password file. However, since the adversary can simply guess a password and test its guess by trying it in the protocol, we equip the simulator with access to a *password verification oracle* $\mathcal{PV}$. The password verification oracle receives queries of the form (u, p) and returns "yes" if $p = spwd_u$ (the secret password of user u), and "no" otherwise.

Let us briefly give some intuition for our definition before spelling out the details. Suppose there is an adversary $\mathcal{A}$ that attacks the system as follows: for a randomly chosen non-compromised user u, it causes u to log in to the system, say, 50 times. It then attempts a single impersonation of u. Suppose $\mathcal{A}$ always succeeds on its first attempt. Clearly, assuming passwords are used at all, this should be a break of the system and our definition should reflect this, so executions of this type should not be simulateable.

However, we know that the simulator, by brute force querying of the password verification oracle, can learn the password with probability one (assuming no intrusion-detection policy). Thus, if we define the simulator naïvely, it can first generate a transcript of 50 simulated successful logins by u and then it can (off the record, using the password verification oracle $\mathcal{PV}$) find $spwd_u$ by brute force. Finally, it can simulate the – successful – impersonation, since it has learned the secret password.

We therefore define a transcript to include certain immutable records of the execution history; these will not be under the control of the simulator. In particular, queries to the password verification oracle, and attempts by non-compromised users to log in, will be recorded even during the simulation, and the simulator will not be permitted to alter these records. Intuitively, this forces the simulator not to access the password oracle $\mathcal{PV}$ with user u more frequently than the adversary tries to impersonate u.

Suppose for pedagogical reasons that

1. Each non-compromised user u is equipped with a special tape. Whenever the adversary causes u to initiate a login attempt, this fact is recorded on the tape.

2. The server is equipped with a special tape. Every time a (possibly impersonated) user attempts to log in, a record of the userid being accessed, together with a bit saying whether or not the attempt was successful, is recorded.

We do not assume the existence of the special tapes in reality.

An *annotated transcript* of an execution includes all the messages exchanged between servers and (compromised and non-compromised) users, together with the contents of the special tapes. The intuition is that if a break occurs it will be reflected in the special tapes: the server's special tape will show more successful accesses to a user's account than attempts recorded on the user's special tape.

The simulator must operate without access to the password file. We assume the existence of a tape-writer $\mathcal{T}$, whose function is to make entries in the special tapes of

the server and the non-compromised users upon *legitimate* requests by the simulator.

There are no real non-compromised users in the simulation: messages supposedly sent by non-compromised users must be generated by the simulator, who does not know their passwords. However, the simulator has access to the passwords of compromised users, since these are known to the adversary.

Whenever the simulator accesses the password verification oracle $\mathcal{PV}$ and receives a "yes/no" reply, $\mathcal{T}$ records the attempt and the outcome on the server's special tape.

Whenever the adversary causes a (simulated) user u (which is non-compromised) to attempt to log in, $\mathcal{T}$ records the attempt on (the simulated) u's special tape.

The simulator may request $\mathcal{T}$ to record as successful, on the server's special tape, an attempt by a non-compromised user. However, for every such attempt there must be a corresponding invocation on the user's special tape. The simulator is bound by the intrusion detection policy, which may decide to fail the attempt anyway, and in this case a "no" is recorded.

The simulator may request $\mathcal{T}$ to record an unsuccessful attempt to log in to the account of a non-compromised user u. There needn't be any corresponding invocation on u's special tape.

Returning to our example with the 50 observations: under our constraints every failed guess by the simulator of a non-compromised user's password is recorded as "part of the simulation," so the transcript of any simulated execution involving off-line brute-force guessing will always be distinguished from the transcript of a real execution in which the adversary, having seen 50 real password verifications, succeeds at its first attempted impersonation.

Definition 4.1 *A protocol is secure against a given adversary class if for all joint distributions $\mathcal{D}$ and all adversaries $\mathcal{A}$ in the given class there is a probabilistic polynomial time transcript simulator $\mathcal{S}$ that interacts with $\mathcal{A}$, a password verification oracle $\mathcal{PV}$, and a tape-writer $\mathcal{T}$ as described above, such that annotated transcripts of real long-lived runs with adversary $\mathcal{A}$ are probabilistic polynomial time indistinguishable from those produced by $\mathcal{S}$.*

5 Solutions for the Multi-User Case

There are several ways to amend the Halevi-Krawczyk approach so that it will function securely in a multi-user environment and satisfy Definition 4.1:

1. Strengthen the security of the encryption scheme by making it non-malleable to chosen-ciphertext postprocessing attacks (the adversary knows the challenge ciphertext and can repeatedly and adaptively ask the decryption mechanism to decrypt any ciphertext except the challenge ciphertext). This is a natural suggestion given that, as discussed in the Introduction, the issue is one of malleability (see also Section 7). The recent Cramer-Shoup cryptosystem [10] makes this approach attractive.
2. Have the server select and assign to each user u a "personal" public encryption key to be used by u when sending messages to the server; this key will be used only by u and not by any other user. The cryptosystem used for generating these keys should be non-malleable with respect to chosen plaintext attacks, or at least semantically secure under one-ciphertext verification attacks. This approach is cumbersome and may require the server to sign each personal public key.
3. Have the server and the user perform a fresh (secure against replay) authenticated secret key exchange, where the secret key that is exchanged is the user's password and authentication is assisted using the server's public signature key.

We will give a complete proof for Suggestion 1 below. However, we first briefly expand on Suggestion 3, since in Section 5.2 we show how to strengthen the requirements on the public-key cryptosystem in order to achieve security against a *dynamic* adversary who chooses which users to compromise as a function of the history of the run. Handling a dynamic adversary is always challenging. In particular, once a process is compromised all secret information regarding its previously encrypted messages becomes available to the adversary. This is a common assumption to make, as it is dangerous to assume that "erasure" of state is possible. This is particularly so in the scenario we are dealing with in this paper, where the users do not have complete control over the computers they are using.

We have in mind an adaptation of a system proposed by Dolev, Dwork and Naor [11] (see Section 3.6 of the full version of the paper). The idea for the system is straightforward: for each interaction the server chooses a fresh (public key, private key) pair that is used only for one message. However, this is not sufficient, since an active adversary may intercept the keys and substitute its own keys. We prevent this behavior by using signatures by both parties. As in the [17] scenario, the server will have a single fixed public key, a hash of which is given to the user at set-up time. However, here the server's public key will be used for signing and not for encryption. The signature scheme used should be *existentially unforgeable*, i.e. an adversary should not be able to generate a valid-looking signature on any message not explicitly signed by the legitimate signer. See [14] for exact definitions. The user will choose a "session" key, which will be used to sign a single message at the third round. Therefore a *one-time* type signature scheme suffices. A user u with password $spwd_u$ wishing to be authenticated by server i whose public key for signing is P_i, performs the following protocol with server i:

1. User u chooses a *fresh* private/public pair of signature keys (s_u, p_u) and sends the public part, p_u, together with his name u to i (lower case is used to distinguish p_u from P_i).

2. Server i chooses a *fresh* private/public pair of *encryption and decryption* keys (E_{iu}, D_{iu}), where E_{iu} is *semantically secure against chosen plaintext attack*, and sends E_{iu} together with $S_i(E_{iu} \circ p_u)$ (*i.e.* a signature on the fresh public-key E_{iu} concatenated with the public signature key u chose) to u; u verifies the signature and that p_u is indeed the public key it chose in Step 1.

3. User u encrypts $spwd_u$ using E_{iu} and sends $E_{iu}(spwd_u)$ together with $s_u(E_{iu}(spwd_u))$ to i. Server i verifies that the message encrypted with E_{iu} is indeed signed with the corresponding p_u.

Note that the sender's key pair (s_j, p_j) in Step 1 is for a one-time signature scheme, which can be implemented using a few (say 100) evaluations of a one-way function. The

server may use a signature scheme such as in [9, 12], which makes the whole approach is relatively efficient. We will return to this scheme in Section 5.2, where we give an implementation of the cryptosystem that yields a password authentication scheme resilient to a *dynamic* adversary.

5.1 Proof of Security

The proofs of security in the multi-user case for of all of the three approaches are similar. We describe in detail the proof for Suggestion 1.

Let f be the concatenation function. Halevi and Krawczyk don't specify what is to be done when a user fails to respond correctly to a challenge. We will assume that when the server issues a challenge r to a user u, the server adds the *challenge pair* (u, r) to a list of challenges. A challenge (u, r) that has not been successfully answered is said to be *outstanding* for u. When a server accepts a response as valid, it informs the user of which challenge was successfully answered and removes the challenge pair from the list of outstanding challenges.

5.1.1 Outline of The Simulator

The simulator will choose a secret/public pair of keys S, P for the simulated server. The non-compromised users will never actually send any messages. The simulator will choose a fixed random string σ. Whenever the adversary $\mathcal{A}$ schedules such a user to respond to a challenge, the simulator will send a random encryption of σ.

As explained in Section 4, the simulator has access to a password verification oracle $\mathcal{PV}$ that knows the passwords of all the parties; the oracle will be consulted *only* on messages generated (not copied) by the compromised users (who may be attempting to impersonate non-compromised users).

If $\mathcal{A}$ causes a non-compromised user u to attempt to log in, then $\mathcal{T}$ records this attempt on u's special tape.

The simulator handles ciphertexts generated by $\mathcal{A}$ differently than ciphertexts generated by non-compromised users. Let c be a ciphertext generated by $\mathcal{A}$. Assume $\mathcal{A}$ causes a compromised user u' to send c to the server as a supposed response on the part of a user u (u may equal u', but need not). The simulator decrypts c and checks that it is an encryption of the concatenation of α, r, u, and S (S is the name of the server), where α is a guess of u's password and r is some outstanding challenge for u. If the plaintext is not of this form, the simulator asks $\mathcal{T}$ to record on the server's special tape the fact that an unsuccessful attempt to access u's account has taken place. If the plaintext is of this form, then the simulator gives the pair (u, α) to $\mathcal{PV}$, whether or not u is compromised. This attempt, and the outcome, is recorded by $\mathcal{T}$ on the server's special tape.

Let c be a ciphertext generated by a simulated non-compromised u. In this case, accessing $\mathcal{PV}$ makes no sense, since the simulation never has access to the password of u, so the simulator must decide what to do according to the context in which c is sent to the simulated server.

- If $\mathcal{A}$ causes c to be sent to the server as a response by u to a challenge from the server, then the simulator requests $\mathcal{T}$ to record a successful access by non-compromised user u on the server's special tape.
- If $\mathcal{A}$ causes c to be sent to the server as a response to a challenge by some compromised party trying to log in as any $u' \neq u$, then the simulator requests $\mathcal{T}$ to record an unsuccessful access by u' on the server's special tape. In this case, if this were not a simulation then c would be an encryption of $(spwd_u, r, u, S)$; in contrast, any valid response that $u' \neq u$ would send should have u' as the third component. Thus, if this were not a simulation then capturing c and re-sending it as a response by u' would unconditionally result in failure.
- If $\mathcal{A}$ causes c to be sent to the server, as a response to a challenge by some compromised party u' trying to impersonate u, then the simulator requests $\mathcal{T}$ to record success if and only if the simulator, in generating c, was simulating a response by u to an outstanding challenge r, and r is still an outstanding challenge for u. In case of success, the challenge pair (u, r) is removed from the list of outstanding challenges. This rule captures the clear-channel case. If instead c was generated in response to a challenge r' that was generated by $\mathcal{A}$ and sent to u from some compromised u' pretending to be the server (so in particular (u, r') is not on the list of outstanding challenges), then the simulator requests $\mathcal{T}$ to record failure.)

Finally, recall that intrusion detection is performed by $\mathcal{T}$, and thus $\mathcal{T}$ complies with the simulator's requests only in so far as they do not conflict with the policy.

5.1.2 Indistinguishability of Simulated and Actual Transcripts

For the sake of contradiction, let $\mathcal{A}$ be a forger as described in Section 4.1, for which no polynomial-time simulator satisfying Definition 4.1 exists. In particular, there exists a polynomial-time bounded distinguisher D that distinguishes transcripts generated by the simulator outlined in Section 5.1.1 and (annotated) transcripts generated in the real system. Let $Q(k)$ denote the running time of D on security parameter k, so that $Q(k)$ is an upper bound on the length of the transcripts we need to consider. Then there exists a polynomial $\Pi(k)$ such that for infinitely many values of k

$$|\Pr[D(simulated) = 1] - \Pr[D(real) = 1]| \geq \frac{1}{\Pi(k)}$$

Using $\mathcal{A}$ as an oracle, we will design an adversary $\mathcal{B}$ that breaks the non-malleability of the underlying cryptosystem under chosen-ciphertext attacks. (The construction is similar to the construction of the adversary A in the proof of Theorem 2.1, given in [17], and is essentially a hybrid argument.)

$\mathcal{B}$ receives as input a public key chosen according to the public-key cryptosystem generator. $\mathcal{B}$ then runs the adversary $\mathcal{A}$ to choose which processors will be compromised, is given passwords for the non-compromised users, and allows the adversary to choose passwords for the compromised users. $\mathcal{B}$ learns the passwords of the compromised users. Before proceeding with the description of $\mathcal{B}$ we define several types of transcripts.

Annotated Real Transcripts. *Annotated* real transcripts, defined in Section 4.2, are real transcripts that are annotated with the contents of the special tapes:

1. the tape of a non-compromised user u records each initiated login attempt;
2. the tape of the server records each attempted login and success or failure of the attempt.

The annotations simply record the events that actually occurred during the real execution.

$real_{Q(k)}$. These are identical to annotated real transcripts. The only difference is the way in which they are generated: $\mathcal{B}$ plays the role(s) of all the non-compromised users. There is a real server involved, who has chosen the cryptosystem.

$real_\ell$, **for** $\ell \in \{1, \ldots, Q(k)\}$. These are a hybrid between $real_{Q(k)}$ and transcripts $real_0$ (which we will show to be distributed exactly the same as simulated transcripts). The first ℓ challenges to non-compromised users u are answered correctly by $\mathcal{B}$. After that, every reply by a non-compromised user u to an outstanding challenge r is "apparently" answered with an encryption of a random string σ, in the sense that this is the ciphertext seen by $\mathcal{A}$. If $\mathcal{A}$ allows the ciphertext to go through as u's reply to the outstanding challenge r, then an adaptor, working with $\mathcal{B}$ (who knows all the passwords), sends to the server an encryption of the proper response.

$real_0$. These transcripts are a special case of $real_\ell$, in which no challenges to non-compromised users are answered with encryptions of proper replies, but rather all are answered with encryptions of a randomly chosen value σ. It is straightforward to show that transcripts of this type are identical to simulated transcripts; the key point is that in both cases the adversary $\mathcal{A}$ is *only* given access to encryptions of a random string σ and *never* given access to encryptions of any string containing the passwords of non-compromised users.

We therefore have by assumption on $\mathcal{A}$ that for some $\ell \in \{1, \ldots, Q(k)\}$

$$|\Pr[D(real_{\ell-1}) = 1] - \Pr[D(real_\ell) = 1]| \geq \frac{1}{\Pi(k)Q(k)}$$

Assume for simplicity that such an ℓ is known to $\mathcal{B}$. The system is run against $\mathcal{A}$ until the ℓth reply to a challenge by a non-compromised user is about to be issued. Let the challenge string be r_ℓ and let u be the user about to make the reply. The message space on which $\mathcal{B}$ will choose to be tested will be $\{f(spwd_u; S, u, r_\ell), \sigma\}$. At this point, $\mathcal{B}$ will be given a ciphertext challenge c that is either an encryption of σ or an encryption of $f(spwd_u; S, u, r_\ell)$. During the remainder of the run of the password authentication protocol, all challenges to non-compromised users will receive as replies encryptions of σ, as in $real_\ell$. If the ciphertext challenge is an encryption of σ, then the transcript is an instance of $real_{\ell-1}$, otherwise it is an instance of $real_\ell$. $\mathcal{B}$ presents the transcript to D. $\mathcal{B}$ outputs D's response. We therefore have:

Theorem 5.1 *Let $\mathcal{E}$ be a cryptosystem that is non-malleable against an adaptive chosen ciphertext attack in the post-processing mode. Then $\mathcal{G}(\mathcal{E})$ satisfies Definition 4.1 against any forging adversary $\mathcal{A}$ as described in Section 4.1.*

5.2 Dynamic Corruption of Users

We now return to Suggestion 3 for the multi-user case, in which an authenticated secret key exchange is performed on the user's password. By using a special encryption scheme, which can withstand dynamic corruption of users by the adversary, we obtain a password authentication scheme that is itself resilient to dynamic adversaries.

The use of the special encryption schemes is critical, since, otherwise, when a formerly non-compromised user u becomes compromised, all of its previously sent messages (*e.g.,* encrypted responses to challenges) become available to the adversary *in the clear*. This is because we do not assume that processors erase the random bits they used in generating their ciphertexts. Thus, if we have been simulating these messages by encrypting a random string σ instead of the appropriate reply to a challenge, the adversary will be able to distinguish transcripts of the simulation from a those of a real execution.

The special encryption scheme provides *non-committing encryption*, a concept introduced by Canetti, Feige, Goldreich, and Naor [8] precisely to tolerate dynamic adversaries in secure multi-party protocols. In such an encryption scheme here are two modes of operation:

normal mode messages are encrypted and decrypted and each ciphertext has a unique decryption.

simulation mode where strings are generated as ciphertexts, but they can be opened (decrypted) in two ways by someone holding the secret keys.

The key point is that encryptions generated in the two modes are indistinguishable.

In [8] the simulator must handle the case in which both the sender and the receiver may be compromised at a later time. In our scenario the problem is easier – permitting more efficient solutions – since the adversary can only compromise the users (the senders of encrypted messages, in our protocol) and not the server (the receiver of the encrypted messages)[2].

In the full paper we describe two non-committing encryption schemes that are sufficiently strong for our scenario. In both schemes, the fact that one can open the bit both ways in the simulation will be used for dynamically simulating the conversation. When a user is compromised it is possible to pretend that before he was corrupted he sent as replies to the server's challenges his *actual* password, even though the simulator didn't know it previously. The schemes are based on trapdoor permutations and on the quadratic residuousity assumption [13], respectively.

6 The Cryptographic Power Necessary for Password Authentication

Over the last fifteen years most cryptographic primitives have been placed in a hierarchy of three equivalence classes, where in each class either all the tasks are possible or none are; it is a hierarchy in the sense that if the higher classes have constructions, then so do the lower ones. The canonical representatives of the three classes (in increasing order) are one-way functions, secret key exchange protocols (SKE), and oblivious transfer (OT).

One-way functions are known to be equivalent to private-key encryption, bit commitment, zero-knowledge proofs, and signature schemes (see Luby's monograph [23]). SKE

[2] In case the server can be compromised as well and the parties are adding secret key exchange and encrypting their communication we will have to either assume that the server can erase random bits used or employ the full [8] scheme. This is sometimes called forward secrecy.

represents the conceptual breakthrough of public-key cryptography. Regarding OT, it is known that if any kind of oblivious transfer is possible, then it is possible to perform general secure function evaluation [21]. The notable result of Impagliazzo and Rudich [20] separates the one-way functions class and secret key exchange *under black-box reductions*. Although no separation between secret key exchange and oblivious transfer is known, the problem of reducing OT to SKE has been open for long time. It is therefore reasonable to conjecture that the two classes may be distinct. In this section, we locate password authentication in the hierarchy.

Halevi and Krawczyk showed that any secure password authentication protocol requires some sort of secret key exchange [17]. Thus, password authentication requires SKE and consequently is above one-way functions in the hierarchy. Our first observation is that secret key exchange is also a sufficient condition for password authentication, *i.e.*: if secret key exchange protocols exist, then so do password authentication protocols. This follows from the fact that our third proposed protocol for password authentication (see Section 5) requires only secret key exchange and a signature scheme. Since secret key exchange implies the existence of signature schemes (this follows from [19, 26, 27]) we have:

Theorem 6.1 *Secure authentication protocols satisfying Definition 4.1 exist if and only if secret key exchange protocols exist.*

Password authentication is often performed in conjunction with secret key exchange in order to encrypt the communication between the parties. In this paper, for the sake of brevity, we considered only "vanilla" authentication, both in our definitions and in our constructions; however, it is relatively simple to extend the discussion to include authenticated secret key exchange (and mutual authenticated key exchange). The extended definition can follow the lines of the work of Bellare and Rogaway in several settings [3, 2, 4]. Extending the constructions accordingly is easy, since, as we have seen, they include secret key exchange anyway.

All the protocols discussed so far in the paper, as well as Definition 4.1, are for the setting where the server has a pair of private/public keys and the public key is known to the the user (or at least can be verified by the user by some other means). It is easy to modify Definition 4.1 to capture the case in which the server does *not* have a public key known to the user. A protocol for performing secure password authentication between two parties in this setting is proposed in [25], based on oblivious evaluation of polynomials[3]. However, in order to use this protocol in a multi-user (and multi-server) environment we must deal with issues of *malleability*, created, for example, by switching the conversations of two users.

One approach is to use non-malleable *commitments* [11]. The [25] protocol consists of two phases: (1) oblivious evaluation of a randomly chosen linear polynomial at a point correpsonding to the password and (2) exchanging parts of the values of the first phase in order to check for consistency. We modify this by adding a non-malleable string commitment to the values that are to be exchanged between the two phases. The issue of identities is solved by concatenating to these values the transcripts of the Phase (1) conversation, and other identifying information.

As it turns out, the use of oblivious transfer is necessary in this case, and secret key exchange protocols are not sufficient. The proof of necessity is based on showing that such a protocol can be used to perform an oblivious evaluation of the "or" function. The latter is known to yield Oblivious Transfer [22].

Theorem 6.2 *Secure authentication protocols satisfying Definition 4.1 without the server having verifiable public keys exist if and only if OT protocols exist.*

7 NM-CPA implies SS-1CV

Theorem 7.1 *If a public-key cryptosystem is non-malleable under chosen plaintext attack then it is semantically secure under one-ciphertext verification attacks.*

For our proof, we use the definition of non-malleability under chosen plaintext attack in [1]. (Under chosen plaintext attack, this is known to be equivalent to non-malleability under the (original) [11] definition [5].) The adversary is split into two parts, A_1 and A_2. A_1 receives the public key drawn according to the pkcs generator. A_1 produces a message distribution M (by providing a sampling algorithm for M), as well as any state information. Next, $x \in_R M$ is chosen, A_2 receives a description of M, the state information produced by A_1, and an encryption $c \in_R E(x)$. A_2 responds with a relation R and a vector $\vec{y}$ of encryptions, where $c \notin \vec{y}$. The vector of decryptions (which can involve "invalid") is denoted $D(\vec{y})$.

Let $\vec{x} = D(\vec{y})$. Let $p_1 = Pr[R(x, \vec{x})]$. Let $p_2 = Pr[R(z, \vec{x})]$, where $z \in_R M$. Then $|p_1 - p_2|$ should be negligible. The probability space is over everything: the choice of the keys, the choices by the adversary, the encryption algorithm, the sampling of M, and any randomness used by R.

Proof. Let (B_1, B_2, B_3) denote an adversary as in Section 2.2. We define a "malleability" adversary (A_1, A_2) accordingly, as follows. A_1 runs B_1 to choose s and then A_1 also chooses $r \in_R \{0,1\}^{|s|}$. Let's assume we are in the high probability case that $r \neq s$. Let M be the uniform distribution on $\{s, r\}$. Choose $x \in_R M$ and give to A_2 the ciphertext $c \in_R E(x)$. A_2 runs B_2 on c to produce x', c'. A_2 outputs the relation $R = R_{(x',c',s,r)}(\alpha, \beta)$, computed in four steps as follows: (1) Let $b = 1$ iff $\beta = x'$, otherwise $b = 0$. (2) B_3 is given b as input, together with the "history" E, s, x', c', c and any other state information for B_1 and B_2. B_3 outputs a guess of whether or not $D(c) = s$. (3) Define g (for "guess") as follows. If B_3 outputs "equal to s" then set $g = s$. Otherwise set $g = r$. (4) If $g = \alpha$ then output 1, else output 0.

Suppose the cryptosystem is non-malleable under chosen plaintext attack. Let M be chosen as described above. Fix $x \in_R M$. When $z \in_R M$ and A_2 gets $c \in E(x)$, then since c contains no information about z, A_2's chance of producing x', c' such that $R_{(x',c',s,r)}(z, D(c')) = 1$ is 1/2. Intuitively, this is because $g \in \{s, r\}$ is output in Step 3 of the evaluation of R, independently of z, so we can simply think of z as being chosen uniformly from M *after* g is computed. By the [1] definition of non-malleability, this means that $R_{(x',c',s,r)}(x, D(c'))$ is also (negligibly close to)

[3] a different protocol is suggested by Lucks [24] assuming random oracles.

1/2. So, given $c \in E(x)$, the probability that $g = x$ is negligibly close to 1/2.

The remainder of the proof follows by arithmetic. □

8 On Halevi and Krawczyk's Revised Definitions

In a revision of their original paper, Halevi and Krawczyk attempted to deal with the multi-user case [18]. The revision differs from the original version in two respects:

- The definition of security was changed to incorporate the fact that there are many users and that some of them might be corrupt and cooperate with the adversary.
- The security requirements of the encryption scheme were changed to something resembling an adaptive chosen ciphertext attack.

The revised paper ([18]) does not note the insecurity of the original work ([17]) in the two-user case.

The new definition proposed by Halevi and Krawczyk is weaker than Definition 4.1 proposed in this paper. Intuitively, the weakness comes from the fact that the new definition still concentrates on the fate of a single user, rather than describing security in "system-wide" terms *e.g.*, global system transcripts.

For instance, it is possible to construct an authentication protocol that satisfies the new HK definition yet leaks whether the passwords of two different (non-corrupt) users are equal (omitted for lack of space). Since finding a human-memorizable password is *not* prohibitively difficult, this extra information gives the adversary considerable additional "bang" for its breaking "buck": when the adversary attacks a system, it can first find a large clique of users with a common password, and then use the remaining queries in exhaustively searching for this password. On average, such an adversary can break into many more users' accounts than an adversary that simply performs the same number of queries using a password verification oracle. Thus, the revised definition of [18] does not adequately model the security requirement of a multi-user environment.

References

[1] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations among notions of security for public-key encryption schemes*, Advances in Cryptology - Crypto'98, LNCS vol. 1462, Springer, 1998, pp. 26–45.

[2] M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption – How to Encrypt with RSA*, Advances in Cryptology – Eurocrypt'94, LNCS vol. 950, Springer-Verlag, 1994, pp. 92-111.

[3] M. Bellare and P. Rogaway, *Entity Authentication and key distribution*, Advances in Cryptology – Crypto'93, LNCS vol. 773, Springer-Verlag, 1994, pp. 232–249.

[4] M. Bellare and P. Rogaway, *Provably Secure Session Key Distribution: The Three Party Case*, Proc. 27th Annual ACM Symp. on the Theory of Computing, 1995, pp. 57–66.

[5] M. Bellare and A. Sahai, *Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization, to appear*, Proc. CRYPTO'99.

[6] S.M. Bellovin and M. Merritt, *Encrypted key exchange: Password-based protocols secure against dictionary attacks*, Proc. of the 1992 IEEE Computer Society Conference on Research in Security and Privacy, pages 72-84, 1992.

[7] R. Canetti, *Security and Composition of Multi-party Cryptographic Protocols*, Theory of Cryptography Library: Record 98-18. Available: `http://philby.ucsd.edu/cryptolib.html`

[8] R. Canetti, U. Feige, O. Goldreich, M. Naor, *Adaptively Secure Multi-party Computation*, Proc. of the 27th ACM Symp. on Theory of Computing, 1996, pp. 639–648.

[9] R. Cramer and I. Damgard, *New generation of secure and practical RSA-based signatures* Advances in Cryptology - Crypto '96, LNCS vol. 1109, Springer, 1996, pp. 173-185.

[10] R. Cramer and V. Shoup, *A practical Public Key Cryptosystem Provable Secure against Adaptive Chosen Ciphertext Attack*, Advances in Cryptology - Crypto'98 LNCS vol. 1462, Springer, 1998, pp. 13–25.

[11] D. Dolev, C. Dwork and M. Naor, *Non-Malleable cryptography*, Preliminary version: Proc. of the Twenty third ACM Symposium on Theory of Computing, 1991, pp. 542–552. Full version, to appear, Siam J. on Computing. Available: http://www.wisdom.weizmann.ac.il/~naor/onpub.html

[12] C. Dwork and M. Naor, *An Efficient Existentially Unforgeable Signature Scheme and its Applications*, Journal of Cryptology, vol 11, 1998, pp. 187-208.

[13] S. Goldwasser and S. Micali, *Probabilistic Encryption*, J. Com. Sys. Sci. 28 (1984), pp 270-299.

[14] S. Goldwasser, S. Micali and R. Rivest, *A Secure Digital Signature Scheme* , Siam Journal on Computing, Vol. 17, 1988, pp. 281-308.

[15] L. Gong. Efficient network authentication protocols: Lower bounds and optimal implementations. Distributed Computing, 9(3):131-145, 1995.

[16] L. Gong, M.A. Lomas, R. Needham, and J. Saltzer. Protecting poorly chosen secrets from guessing attacks. IEEE Journal on Selected Areas in Communications, 11(5):648-656, June 1993.

[17] S. Halevi and H. Krawczyk, *Public-key Cryptography and password protocols*, 5th ACM Conference on Computer and Communication security, pp. 122–131, 1998.

[18] S. Halevi and H. Krawczyk, *Public-key Cryptography and password protocols*, Theory of Cryptography Library: Record 99-04, 1999. Available: http://philby.ucsd.edu/cryptolib.html

[19] R. Impagliazzo and M. Luby, *One-way functions are essential to computational based cryptography*, Proc. of the 30th IEEE Symp. on the Foundation of Computer Science, 1989, pp. 230–235.

[20] R. Impagliazzo and S. Rudich, *On the Limitations of certain One-Way Permutations* , Proc. 20th ACM Symp. on Theory of Computing, 1989, pp. 44–61.

[21] J. Kilian, **Use of Randomness in Algorithms and Protocols**, MIT Press, Cambridge, Massachusetts, 1990.

[22] J. Kilian, *A general completeness theorem for two-party games*, Proc. 23rd ACM Symp. on Theory of Computing, 1991, pp. 553–560.

[23] M. Luby, **Pseudo-randomness and applications**, Princeton University Press, 1996.

[24] S. Lucks, *Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys*, Proc. of Security Protocol Workshop '97, `http://www.dmi.ens.fr/~vaudenay/spw97/spw97_Luc3.ps.gz`.

[25] M. Naor and B. Pinkas, *Oblivious Transfer and Polynomial Evaluation* Proc. 30th ACM Symp. on Theory of Computing, 1999, pp. 245–254.

[26] M. Naor and M. Yung, *Universal One-way Hash Functions and their Cryptographic Applications*, Proc. 21st ACM Symp. on the Theory of Computing, 1989, pp. 33–43.

[27] J. Rompel, *One-way Functions are Necessary and Sufficient for Signatures*, Proc. 22nd Annual ACM Symposium on the Theory of Computing, Baltimore, 1990, pp. 387–394.

[28] T. Wu, *The Secure Remote Password Protocol*, Proc. of the 1998 Internet Society Network and Distributed System Security Symposium, San Diego, March 1998, pp. 97-111.