Open Key Exchange:
How to Defeat Dictionary Attacks Without Encrypting Public Keys

Stefan Lucks

Institut für Numerische und Angewandte Mathematik
Georg–August–Universität Göttingen
Lotzestr. 16–18, D–37083 Göttingen, Germany
(email: `lucks@math.uni-goettingen.de`)

Abstract. Classical cryptographic protocols based on shared secret keys often are vulnerable to key-guessing attacks. For security, the keys must be strong, difficult to memorize for humans. Bellovin and Merritt [4] proposed "encrypted key exchange" (EKE) protocols, to frustrate key-guessing attacks. EKE requires the use of asymmetric cryptosystems and is based on encrypting the public key, using a symmetric cipher.
In this paper, a novel way of key exchange is presented, where public keys are sent openly, not encrypted. In contrast to EKE protocols, the same public-key/secret-key pair can be used for arbitrary many protocol executions. The RSA-based protocol variant is found to be quite efficient and practical.
Compared to previous work on such protocols, a more solid formal treatment is given, influenced by the work of Bellare and Rogaway [3] on key exchange protocols for *strong* common secrets.

1 Introduction

Most humans, including the current author, are quite bad at memorizing large randomly generated keys. In the real world, many people choose weak passwords or use a short PIN ("personal identification number"). Hence, there is a real need for cryptographic protocols to give a reasonable degree of security even if rather weak keys are used.

In 1992, Bellovin and Merritt [4] (see also section 22.5 of [12]) addressed this problem. Their solution was an "encrypted key-exchange" (EKE) protocol. The core idea behind EKE is to use the secret (symmetric) key π to encrypt the public key E of a randomly chosen asymmetric key pair (E, D), then to use E to encrypt the randomly chosen (symmetric) session key k and finally to decrypt $E(k)$ using D. The security of EKE depends on the fact that two parties using different values for π are likely to use different public keys E and E', and that knowing an asymmetric key pair (E', D') does not help to decrypt $E(k)$. It is important for EKE protocols, that every public key is used only once.

This paper's approach is somewhat different from EKE. Though we also make use of public key cryptography, the public key is not encrypted but sent openly in the clear. Also, the public key may be reused arbitrarily often.

2 Our Model of Communication

In this section[1], we formalize how Alice and Bob exchange keys while the adversary Eve controls the communication network. Our model is inspired by the work of Bellare and Rogaway [3]. At her will, Eve can deliver messages between Alice and Bob. She is able

- to read all messages between both parties,
- to modify such messages, to delay them or to send them multiple times,
- and to send messages generated by herself to Alice or Bob.

Also, Eve can create several instances of Alice and Bob and communicate with these instances in parallel sessions. Both Alice and Bob are represented by an infinite collection of oracles. These oracles can be called by Eve and then change their internal state and compute an output, but they never directly communicate with each other. We denote the oracles by Π_A^i resp. Π_B^j. A and B indicate "Alice" resp. "Bob" while i and $j \in \mathbb{N}$ indicate the instance ($\mathbb{N} = \{1, 2, \ldots\}$).

The functions Π_A and Π_B define the behavior of Alice and Bob. For $I \in \{A, B\}$, the functions Π_I takes the following inputs:

- 1^K (the security parameter $K \in \mathbb{N}$ in unary notation),
- $\pi \in \{0, 1\}^*$ (the secret information),
- $c \in \{0, 1, -\}^*$ (the communication so far; "$-$" is used as a separator),
- and $r \in \{0, 1\}^\infty$ (an infinite amount of fair private coin flips),

to compute the outputs pair (x, y) with

- $x \in \{0, 1\}^*$ (the next message to send—the "reply")
- and $y \in \{\{0, 1\}^*, \text{reject}, \text{none}\}$.

The oracle can accept a communication, then $y \in \{0, 1\}^*$ is the session key. Also, the oracle can reject a communication or refuse to make a decision at that point of time. If it accepts or rejects, the decision is final.

Running a protocol in the presence of Eve, given a security parameter K, means applying the following experiment:

1. Choose a secret π.
2. For $I \in \{A, B\}$ and $j \in \mathbb{N}$, initialize the $c_{I,j} \in \{0, 1\}^*$ by the empty string.
3. For $I \in \{A, B\}$ and $j \in \mathbb{N}$, choose a random string $r_{I,j} \in \{0, 1\}^\infty$.
4. Run the adversary Eve, answering oracle calls as follows. When Eve asks a query $(I, j, z) \in \{A, B\} \times \mathbb{N} \times \{0, 1\}^*$, oracle Π_I^j
   - replaces $c_{I,j}$ by the concatenation of $c_{I,j}$, "$-$", and z,
   - computes $(x, y) := \Pi_I\left(1^K, \pi, c_{I,j}, r_{I,j}\right)$,
   - replies x,
   - and tells Eve, whether it accepted, rejected, or did neither.

[1] Readers with little interest in the formal treatment of cryptographic protocols may skip ahead to the next section (and possibly go back to this section later).

Once the oracle Π_I^j has accepted, Eve can ask Π_I^j for its session key.

5. Finally, Eve chooses an oracle Π_J^i and attempts to guess its session key.

If Eve chooses a pair of oracles Π_A^i and Π_B^j and then faithfully conveys each flow from one oracle to the other, then we call the oracles Π_A^i and Π_B^j **related**. If two oracles are related, Eve did behave like a **benign adversary**[2].

The adversary Eve is **successful**, if it guesses the the session key of an accepting oracle Π_I^j without having asked either Π_I^j or a related oracle for its session key. Note that a benign adversary still can either ask accepting oracles for their session keys or attempt to guess their session keys.

In addition to the above, Alice and Bob, or their instances, reject after waiting a reasonable amount of time for an answer from their partner (time-out).

Now, at last, we can define what we are talking about in this paper:

Definition. Let π be randomly chosen from the set S_π of size S, according to the uniform probability distribution. We say that the protocol defined by the pair of functions Π_A and Π_B is a **secure key exchange protocols for weak common secrets**[3],

D1: if two oracles are related, then with overwhelming probability both accept and compute identical session keys,

D2: if the probability of success is negligible for every efficient benign adversary,

D3: and if, after Eve has been rejected R times, her probability of success without being rejected again does not significantly exceed $1/(S-R)$.

Condition D1 says, if each party's messages are faithfully relayed to one another, then the parties succeed in finding a common session key—at least with overwhelming probability. Due to condition D2, passive attacks are hopeless. Condition D3 appears somewhat complicated, but it simply means that active attacks cannot significantly improve on the trivial attack. The trivial attack consists of Eve choosing any value $\psi \in S_\pi$ for π and then masquerading as either Alice or Bob. If the other party accepts, Eve concludes that her guess was correct. If the other party rejects, Eve knows $\psi \neq \pi$ and the next time chooses $\psi' \neq \psi$ for π. At first, Eve's probability of success is $1/S$, and after $R < S$ failed attacks it has grown to $1/(S-R)$.

We could generalize the above definition such that π is randomly chosen from the set S_π, according to *any probability distribution*, as long as for every element ψ of S_π the probability $\text{prob}(\pi = \psi)$ is bounded by $\text{prob}(\pi = \psi) \leq 1/S$. For the sake of simplicity, we restrict ourselves to the uniform probability distribution.

[2] This notion is due to Bellare and Rogaway [3]. If our key exchange protocol is any good, two related oracles both accept and their session keys are equal—at least with overwhelming probability.

[3] In the sequel, we sometimes simply use "password" instead of "weak common secret".

3 The generic open key exchange (OKE) protocol

In this section, we describe the generic open key exchange protocol, define its security requirements and prove it to be secure.

Our notation and requirements can be described like this:

> Let K be our security parameter, large enough that brute-forcing 2^K values is infeasible, the number S of choices for the secret small enough that brute-forcing S^2 values is still feasible (hence $S \ll 2^K$), and $l \geq 2K$. For the protocol, we need hash functions $h_i : \{0,1\}^* \longrightarrow \{0,1\}^l$ ($i \in \{1,2,3\}$), an asymmetric key pair (E, D), such that E describes the encryption operation $E : S_E \longrightarrow S_E$, where S_E is a finite group "$\diamond$" denotes the group operation and "$\neg$" its inverse, and a hash function $H : \{0,1\}^* \longrightarrow S_E$. The group operation needs to be compatible with the encryption function, i.e. for all $x, y \in S_E$: $E(x) \diamond E(y) = E(x \diamond y)$. Also, the encryption function must be invertible, i.e. $D(E(x)) = x$ must hold for $x \in S_E$.
> The group S_E can depend on the choice of E, as for RSA. The pair (E, D) and hence the group S_E can be chosen once by Alice and remains fixed during the experiment of running the protocol in the presence of Eve. D is kept secret. We use "$|$" to indicate the concatenation of bit strings.

Our protocol works like this:

0. *Both in advance:* Alice and Bob agree on a common secret $\pi \in_{\mathrm{R}} \{1, \ldots, S\}$.
   Alice in advance: Alice generates a public-key/secret-key pair (E, D).
1. Alice chooses $m \in_{\mathrm{R}} \{0,1\}^l$ and sends E and m to Bob.[4]
2. Bob chooses $\mu \in_{\mathrm{R}} \{0,1\}^l$ and $a \in_{\mathrm{R}} S_E$, computes $p = H(E|m|\mu|\pi)$ and $q = E(a) \diamond p$, and sends μ and q to Alice.
3. Alice computes p like Bob, $a' = D(q \neg p)$, and sends $r = h_1(a')$ to Bob. (Now $a' = a$.)
4. If $r \neq h_1(a)$, Bob rejects. Otherwise, he computes $k = h_2(a)$ and $t = h_3(a)$, sends t to Alice, and uses k as his session key.
5. Alice rejects if $t \neq h_3(a')$. Else, she uses $k' = h_2(a')$ as her session key.

We make the following two cryptographic assumptions:

A1: The hash functions H, h_1, h_2, and h_3 behave like independent random functions. In the notion of Bellare and Rogaway [2], the functions h_i are public "random oracles", i.e. Eve can send queries $x_1, x_2, \ldots$ to the random oracle h_i resp. H and receives answers $h_i(x_j)$, resp. $H(x_j)$, all independent random values. Only when a query $x_c = x_j$ is sent to the oracle h_i (or H) and Eve was given the oracle response $h_i(x_j)$ ($H(x_j)$) before, the result is predictable: $h_i(x_c) = h_i(x_j)$ (or $H(x_c) = H(x_j)$).

[4] Note that though the pair (E, D) can be chosen once, it does not make sense to assume Bob can remember E. If Bob already did know Alice's authentic public key E, he could simply choose the session key k_B and send $E(k_B)$ to Alice, instead of runing the protocol.

A2: Given a value x, randomly chosen from the set S_E, it is infeasible to find a value y with $y = D(x)$.

Note that A1 can actually be satisfied by using one single random function $h : \{0,1\}^* \longrightarrow \{0,1\}^l$, defining $h_1(x) = h(01|x)$, $h_2(x) = h(10|x)$, $h_3(x) = h(11|x)$, and H depending on the values $h(0000\dots00|x), h(0000\dots01|x), \dots, h(00\dots|x)$.

Theorem 1. *Under the assumptions A1 and A2, the generic OKE protocol is a secure key exchange protocol for weak common secrets.*

Throughout the proof, "essentially" means "with overwhelming probability".

Proof. One can easily verify that condition D1 holds.

The values m, μ, and a are independent random values from $\{0,1\}^l$ resp. S_e. Due to $D(E(a)) = a$, the attacker gets no information about p from $q = E(a)\Diamond p$.

Given μ and q, Alice replies $r = h_1(a')$. If Eve did guess a', she could use r to check the correctness of this guess—and then compute π. But finding a' is equivalent to solving the equation $a' = D(q\neg p)$ and hence to decrypting a random ciphertext—even if p is known. Due to assumption A2, this is infeasible. By definition, the value $r = h_1(a')$ is just a random value for everyone not knowing a'. The same holds for $t = h_3(a)$. In other words, as long as Eve remains passive, all oracle replies look like random values from $\{0,1\}^l$ resp. from S_E for her—essentially. Thus, D2 is satisfied.

We still have to proof D3. Due to $l \geq 2K$, all instances Alice and Bob will create new values m, μ, and a, essentially.[5] This rules out replay attacks, where Eve waits for Alice to repeat herself to repeat Bobs responses, or for Bob to repeat himself, to repeat Alices responses.

Note that after starting an active attack, Eve must give a correct answer to avoid being rejected.

At first, we consider the case $R = 0$, i.e. no oracle did reject before when Eve attempts to guess a session key. Depending on the protocol step when Eve first becomes active, we argue:

1st step: If Eve actively chooses a pair (E, m) and sends it to Bob, she has to choose a response $r = h_1(a')$ in the third step with $a' = D(q\neg p)$. There are only S possible values p, thus there are essentially S different responses r. If Eve's choice is wrong, Bob rejects and Eve may be able to delete one of the S possibilities for p from her list. The probability for Eve's choice to be correct is $1/S$.

2nd step: If Eve, after having seen a pair (E, m) from an instance of Alice, actively chooses μ and q as a response, she receives the value $r = h_1(D(q\neg p))$ from Alice—and in principle knows enough to verify which value p is used

[5] We required K to be large enough that computing 2^K values is infeasible. But due to the birthday paradox, we'd need to call our oracles about $2^{l/2} \geq 2^K$ times to get one value m or μ which was used before.

by Alice. But due to assumption A1, the value r does Eve not help with computing a, other than as a verifyer whether the correct a has been found. Let Eve be able to find values a_1, p_1, a_2, p_2, and q with $p_1 \neq p_2$ two different candidates for p and

$$E(a_1)\diamond p_1 = q = E(a_2)\diamond p_2. \tag{1}$$

This means, she can find a plaintext $a_\Delta = a_1 \neg a_2$ with

$$E(a_\Delta) = p_2 \neg p_1,$$

and thus is able to decrypt the difference $p_2 \neg p_1$ of two random values. Since S is "small" and there are only S^2 such differences, this means that there is an efficient probabilistic algorithm for Eve to decrypt random values, in violation of assumption A2.

Thus, Eve can't know more than one candidate for a and hence not check more than one candidate for p. If her candidate for a is wrong, she can't find the correct answer t for step 4 and will cause Alice to reject in step 5, essentially. Again, the probability for Eve's choice to be correct is $1/S$.

Later: If Eve acts like a benign adversary in the first two steps in order to become active in the third or fourth step, it is infeasible for her to compute either a or the correct response values $r = h_1(a)$ or $t = h_3(a)$, essentially. Then her probability of success is negligible, i.e. $\ll 1/S$.

Hence, D3 holds for $R = 0$. Note that by excluding one candidate p_i from the set of possible candidates for p, we also exclude the corresponding weak secret π_i as a possible choice for π. At first there are S candidates $1, \ldots, S$, for π. After R rejections there remain $S - R$ such candidates, w.l.o.g. $1, \ldots, S - R$. By induction we verify that Eve's chance to be correct is $1/(S - R)$, then. □

Note that Alice even may publish her public key E in advance (after the first execution of the protocol Eve knows E anyway) or use the very same public key to communicate with other participants than Bob, without risking her security.

4 RSA and protected OKE

In this section, we point out a problem with naive RSA-based generic OKE. It comes from Bob not being able to verify all our requirements for OKE to be satisfied. We propose protected OKE, a variant of generic OKE to address this problem.

At a first look, the RSA public-key cryptosystem appears to be ideal for open key exchange. The public key is the pair $E = (e, N)$, where $N = PQ$ is the product of two huge primes P and Q and e is relatively prime to $\varphi(N) = \varphi(P)\varphi(Q) = (P-1)(Q-1)$. The encryption function is defined by $E(x) = x^e \bmod N$, the decryption function is $D(x) = x^d \bmod N$, where the secret exponent d is chosen such that $ed \equiv 1 (\bmod\ \varphi(N))$. The natural choice for the group operation $\diamond$ is the multiplication mod N, the group S_E is the set of all numbers in $\{1,\ldots,N-1\}$ which are not divisible by P or Q, i.e. the size of S_E is $|S_E| = \varphi(N)$.

In the case of RSA, all requirements are easy to verify—with one important exception: The function E must be invertible and hence the equation $D(E(x)) = x$ must hold.[6] In the case of RSA, this is no problem if e and N are chosen according to their specifications—but e and N can be chosen in a different way and Bob is unable to detect this.

We come up with some slightly changed requirements:

1. If chosen by an honest participant (i.e. Alice), the function $E: S_E \longrightarrow S_E$ is invertible.
2. If E is chosen by Eve, the space of the encryption function E can collapse.
3. As before, "$\diamond$" is the group operation of S_E.
4. The finite group S_E is of order $|S_E| \geq 2^{2K}$.
5. $H': \{0,1\}^* \longrightarrow S_E$ is a random function just like H.

All other requirements remain the same.[7]

Since "$\diamond$" is a group operation, there is a number $k \in \{1,2,\ldots,\}$, such that for every $x \in S_E$ the set $E(x)$ has k elements. If $k = 1$, the function E is invertible and we have no problem. Otherwise $k \geq 2$, hence for every ciphertext $y \in S_E$, $y = E(x)$ there are at least two possible decryptions x_1, and x_2, $y = E(x_1) = E(x_2)$.

This observation leads to the **protected OKE protocol**. For protected OKE, we we change generic OKE like this:

> Instead of $\mu \in_{\mathrm{R}} \{0,1\}^l$, Bob chooses 2 values μ_{-1}, $\mu_0 \in_{\mathrm{R}} S_E$. Now Bob computes $\mu_1, \mu_2, \ldots, \mu_K$ by
>
> $$\mu_i = E\left(\mu_{i-2} \diamond H'\left(\mu_{i-1}\right)\right).$$
>
> The last two values μ_{K-1} and μ_K are sent back, instead of μ in the case of the generic protocol.
>
> The value p now is computed by $p = H(E|m|\mu_{-1}|\mu_0|\pi)$.

[6] Otherwise, even if $a \in_{\mathrm{R}} S_E$, the random variable $E(a)$ would not be distributed uniformly, hence Eve could use $q = E(a) \diamond p$ to collect some knowledge about the actual value of p, especially to rule out some of the candidate values for p.

[7] In the case of RSA, all new requirements can easily be enforced by both Bob and Alice. For Bob, enforcing the third requirement means to verify $\gcd(x, N) = 1$ for the RSA-modulus N and all (pseudo-)random values $x \in_{\mathrm{R}} \{1,\ldots,N-1\}$, $x = H(\ldots)$, or $x = H'(\ldots)$ he generates. For practical choices of the security parameters, the fourth requirement is satisfied, except when the RSA-modulus has lots of tiny prime factors. If the order of the group S_E were small, a random value $x \in_{\mathrm{R}} \{1,\ldots,N-1\}$ would likely be $\notin S_E$. Since $x \notin S_E \Longleftrightarrow \gcd(x, N) \neq 1$, checking the gcd is sufficient to protect against violations of the third and fourth requirement.

If E is invertible, the values μ_{-1} and μ_0 can be computed by K evaluations of the decryption function D and the random function H'.

If E is not invertible, given μ_{K-1} and μ_K there are at least two choices μ^1_{K-2}, μ^2_{K-2} for μ_{K-2}, for every μ^i_{K-2} there are two choices for μ_{K-3}, ... and thus we expect 2^K choices $\mu^1_{-1}, \ldots, \mu^{2^K}_{-1}$ for μ_{-1}. Due to the randomness of H' and because $|S_E| \geq 2^{2K}$, the probability that there is a significant number of collisions $\mu^i_{-1} = \mu^j_{-1}$ for $i \neq j$ is negligible.[8] It is infeasible for Eve to compute all possible values for p, even if π is fixed. Thus, the partial knowledge about p she gets from inspecting $q = E(a)\Diamond p$ is useless for her.

This way, we have modified our protocol such that it is secure even if Bob can't directly check E to be invertible.

For a secure realization of protected RSA-OKE, the following choices are suggested: $K = 80$ for the security parameter K, $N \approx 2^{1000}$ for the modulus N and the use of a dedicated hash function like SHA-1 with 160 bits of output (or more) as the underlying primitive to realize pseudorandom functions H, H', h_1, h_2, and h_3. For the public encryption exponent e we may make a fixed choice which allows speedy RSA encryption, e.g. $e = 3$.

With these parameters, the communication costs are quite reasonable: Four 1000-bit numbers N, μ_{-1}, μ_0, and q and three 160-bit strings m, r, and t are exchanged between Alice and Bob.

5 A comparison between OKE and EKE protocols

In this section we compare EKE and OKE protocols, with a closer look on their variants based on RSA.

As its inventors write, *encrypted key exchange* (EKE) relies *"on the counter-intuitive notion of using a secret key to encrypt a public key"* [4], with the secret key being derived from the password. If π is a weak secret, known to Alice and Bob, $\pi(X)$ denotes the encryption of X using π. Similarily we write $k(X)$ for encrypting X using the session key k. For an asymmetric key pair (E, D) with the public key E, $E(X)$ denotes the public key encryption of X. The **generic EKE protocol** can be described like this:

1. At random, Alice generates an asymmetric key pair (E, D) and sends $\pi(E)$ to Bob.
2. Bob decrypts this to find the public key E, chooses a random session key k, and sends $\pi(E(k))$ to Alice.
3. Alice decrypts this to get k. Further, she chooses a random challenge c to send $k(c)$ to Bob.
4. Bob decrypts this, chooses a random challenge d, and sends $k(c|d)$ to Alice.
5. Alice checks whether Bobs reply is of the form $k(c|\ldots)$. If so, the sends $k(d)$ to Bob. Else, she rejects.
6. Bob checks whether Alice's last reply is correct and rejects if not.

[8] This argument is based again on the birthday paradox.

Two properties of the EKE are

(A) Every protocol execution requires Alice to generate a new key pair.[9]
(B) The EKE depends on encrypting the public key.

The impact of property (A) is that asymmetric cryptosystems with a slow key-generation process tend to be slow when running the protocol. While e.g. El-Gamal asymmetric encryption and Diffie-Hellman key exchange do not require a lengthy key generation process, cryptosystems like RSA do. RSA requires to find two huge primes. This is no serious problem for OKE protocols, since here, a key generated once can be used forever.

If one considers an RSA-based realization of EKE, things become quite difficult due to property (B). It is hard to see how some RSA-modulus N can be encrypted such that trial decryption using the wrong key is not likely to reveal the fact, that the result of the decryption is not a valid RSA modulus. The inventors of EKE, Bellovin and Merritt, suggest to send N in the clear and only to encrypt the public exponent e [4]. Here, e should be a randomly chosen odd integer. They point out how to choose two huge primes P and Q, such that odd random integers are relatively prime to $\varphi(PQ) = (P-1)(Q-1)$.

While the protected RSA-OKE allows us to choose the public exponent e as we like, e.g. $e = 3$ for fast RSA encryption, RSA-EKE requires e to be an odd random number, which slows down encryption. On the other hand, protected OKE requires Bob to encrypt many (i.e. $K+1$) plaintexts with RSA and Alice to decrypt as much values.

The main advantage of protected RSA-OKE in comparison to RSA-EKE is that public keys are generated once and for all—RSA key generation is very demanding, especially for slow devices like smartcards. This is likely to overcome the main disadvantage of protected RSA-OKE: The need to compute $K+1$ ciphertexts in the case of Bob and to decrypt all these ciphertexts in the case of Alice.

6 Related work

Apart from Bellovin and Merritt [4], only few other key-exchange and authentication protocols for weak common secrets have been designed.

Gong and others [7] presented protocols with the requirement that either authentic public keys had to be known, or had to be exchanged encrypted and

[9] Otherwise, there is a replay attack for Eve: First, she listens to a protocol execution where Alice sends the value $x = \pi(E)$ in the first round and Bob replies $y = \pi(E(k))$ in the second. Then she requests the session key k. If Alice uses $x = \pi(E)$ once again, Eve impersonates Bob and replies y. Now, the session key is k, which Eve knows. Also, if Alice reuses E for communicating with another side, e.g. Dan, using a different password ρ, Eve can find both π and ρ by brute force, looking for two candidates π_i and ρ_j with equal decryptions $\pi_i^{-1}(x) = \rho_j^{-1}(z)$, where $x = \pi(E)$ and $z = \rho(E)$.

remain secret ("secret public key protocols"). For the second variant the authors expressed a concern about finding suitable asymmetric encryption methods. Gong [8] later optimized the protocols of [7]. Among his protocols was a three-round variant.

Another 3-round protocol is due to Steiner, Tsudik and Waidner [13], who optimized the Diffie-Hellman EKE protocol from Bellovin and Merritt.

Jablon [9] gave a very diligent treatment of the Diffie-Hellman EKE protocol and also presented SPEKE, a similar protocol without symmetric encryption.

Information leakage enables passive attackers to eliminate possible passwords. Patel [11] pointed out a possible source of information leakage in EKE protocols. Its effectiveness depends on the structure of the public keys of the asymmetric cryptosystem in use, the blocksize of the symmetric cryptosystem in use, and the padding rules applied when a public key is encrypted using the symmetric cryptosystem. Patel demonstrated that the "random padding" suggested by Bellovin and Merritt [4] can be dangerous.

An approach quite different from EKE-like protocols is "fortified key negotiation", due to Anderson and Lomas [1] (see also section 22.6 of [12]). After exchanging a session key using ordinary public key techniques, a special kind of hash function with many collisions is used to authenticate the session key. This hinders key-guessing attacks, but in general violates our requirement D3 because too much information about the password is leaked.

Bellare and Rogaway [3] presented a model for authenticated key exchange with *strong* common secrets which heavily influenced the model in this paper. At this workshop, Blake-Wilson and Menezes [5] also considered the model of Bellare and Rogaway.

7 Concluding remarks and open problem

For the sake of simplicity, the model in this paper is less general than the one of Bellare and Rogaway [3]. Apart from the fact that attackers can always run the trivial attack with significant probability of success, which is a direct consequence of using a weak common secret, all the restrictions of our model compared to the one of Bellare and Rogaway can be removed.

Simply spoken, a *secure mutual authentication protocol* is a two-party protocol where both parties accept (at least with overwhelming probability) if their adversary is benign, but reject with overwhelming probability in the presence of an active attacker. The formal definition of a secure mutual authentication protocol is lengthy and somewhat complicated, see [3] or Blake-Wilson's and Menezes' paper in these proceedings. Due to Bellare and Rogaway, *secure key exchange* protocols are secure mutual authentication protocols with additional (secure key-transport) properties. It is easy to see that our OKE protocol and its variant are secure mutual authentication protocols, except, of course, that an active attacker's probability of success does not significantly improve on the trivial attack—which is not negligible.

Also, Alice is always the initiator[10], Bob is always the responder, and there is only one initiator and one responder in our model. As was pointed out by Blake-Wilson [6], if the role of initiator and responder could change, Eve could run a replay attack with Alice being both the initiator and the responder. Without extra checks, Alice could accept two sessions seemingly with Bob, both sessions using the same session key. If Eve asked for the one of the session keys, she could successfully "guess" the other. By including the initiator's and the responder's names (or any unique identification strings) into the messages, we can get rid of this restriction, as did Bellare and Rogaway in their protocols.

Due to Jablon [9], *perfect forward secrecy* "means that disclosure of the password doesn't reveal prior recorded conversations". If Eve learns the password π, she can compute the values p of old conversations, but she can't find $a = D(q \neg p)$. On the other hand, disclosure of Alice's secret key E would allow Eve to compute $a_i = D(q \neg p_i)$ with $p_i = H(\ldots|\pi_i)$ for all possible secrets π, to find the value $a_i = a$ by comparing r and $h_1(a_i)$ and thus to decrypt old conversations. This might be a reason for Alice to change her asymmetric key pair (E, D) frequently.

One of our requirements was that the number S of choices for the password was small enough that brute-forcing S^2 values is still feasible. Actually, we made use of this in the context of equation (1). Does this mean that our protocol is insecure if our common secrets are too strong?

Of course not! In the proof we argue that if Eve can find more than one solution for equation (1), she can invert the encryption function E on one out of S^2 ciphertexts. Because this is infeasible for her, she only can exclude one out of S possible choices for the password π.

Now, if S is too large, we can't rule out that Eve can find $s > 1$ solutions for equation (1), exclude s possible choices for π and thus improve on the trivial attack. But this sort of information leakage is very limited, if s were too large, assumption A2 would no longer hold. In other words, as long as the password space is large, we may run the risk of some information leakage, but with decreasing password space the possible information leakage adaptively decreases.

EKE-like protocols seem to be well suited for using public key cryptosystems like Diffie-Hellman and ElGamal. Consequently, most of the recent work on EKE-like protocols [13, 9] concentrates on using Diffie-Hellman key exchange as the underlying public key cryptosystem.

In contrast to this, (protected) OKE protocols seem to be especially well suited for RSA. On the other hand, the ElGamal cryptosystem does not satisfy our requirements, i.e. encrypting with ElGamal can not be described by a function $E : S_E \longrightarrow S_E$. Thus the security proof is of no use here. But is ElGamal-OKE actually insecure?

Yes, it is! Jablon [10] found a simple attack on the generic OKE protocol if ElGamal encryption is used straightforwardly . The attack exploits the fact that ElGamal is a "disclosing encryption system", see end of section 2.4 of [4].

[10] The party which sends the first message in a two-party protocol is called the "initiator", the other party the "responder".

It is an open problem to find more efficient variants of (protected) OKE, e.g. a three round protocol, similar to the optimized protocols found in [8] and [13].

Acknowledgement

The author would like to thank Paul Syverson for providing some relevant references (especially [11]), Simon Blake-Wilson for an interesting discussion on the Bellare-Rogaway model, and David Jablon for many helpful comments on a preliminary version of this paper.

References

1. R. Anderson, M. Lomas, "Fortifying Key Negotiation Schemes with Poorly Chosen Passwords", *Electronics Letters*, Vol. 30, No. 13, 1994, 1040–1041.
2. M. Bellare, P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", *First ACM Conference on Computer and Communications Security*, ACM, 1993.
3. M. Bellare, P. Rogaway, "Entity Authentication and Key Distribution", *Crypto '93*, Springer LNCS 773.
4. S. Bellovin, M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks", *Proc. IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, 1992.[11]
5. S. Blake-Wilson, A. Menezes, "Security Proofs for Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques", 1997, these proceedings.
6. S. Blake-Wilson, private communication.
7. L. Gong, M. Lomas, R. Needham, J. Salzer, "Protecting Poorly Chosen Secrets from Guessing Attacks", *IEEE Journal on Selected Areas in Communications*, Vol. 11, No. 5, 1993, 648–656.
8. L. Gong, "Optimal Authentication Protocols Resistant to Password Guessing Attacks", *Proceedings of the 8th IEEE Computer Security Foundations Workshop*, 1995, 24–29.
9. D. Jablon, "Strong Password-Only Authenticated Key Exchange", *ACM Computer Communications Review*, October 1996.
10. D. Jablon, private communication.
11. S. Patel, "Information Leakage in Encrypted Key Exchange", manuscript.
12. B. Schneier "Applied Cryptography" (2nd ed.), Wiley, 1996.
13. M. Steiner, G. Tsudik, M. Waidner, "Refinement and Extension of Encrypted Key Exchange", *Operating Systems Review*, Vol. 29, No. 3, 22–30.

This paper is to appear in the proceedings of the
1997 Security Protocols Workshop, Springer LNCS

[11] The full version of this paper was found at `http://www-cse.ucsd.edu/users/mihir/papers/eakd.ps.gz`.